مهندس / أسامة الحسيني

تحدث إلى الكومبيوتر

LOGO
بلغة لوجو

لغة أصدقاء الروبوط





مكتبة ابن سينا

سلسلة علمية ثقافية تتناول مختلف العلوم والفنون..

تصدرها

مكتبة القرآن

ويشرف عليها

مهندس / مصطفى عاشور

مهندس / أسامة الحسينى

تحدث إلى الكومبيوتر

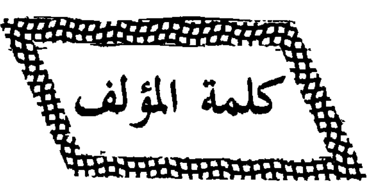
بلغة لوجو LOGO

لغة أصدقاء الروبوط

LOGO
لوجو

- اللغة التى تجعل من الكومبيوتر صديقاً لك .
- لغة التحكم فى الروبوط (الإنسان الآلى) .
- لغة الذكاء الصناعى للكومبيوتر .
- تطبيقات مختلفة على الرسم والألعاب .

مكتبة القرآن
للطبع والنشر والتوزيع
٣ شارع القماش بالفرنساوى - بولاق
القاهرة - ت : ٧٦١٩٦٢ - ٧٦٨٥٩١



● ما هى لوجو ؟

لغة « لوجو » لغة مختلفة تماماً ..

تختلف عن سائر لغات الكومبيوتر فى الشكل والبناء . وتتميز بخصائصها الفريدة فى الرسم علاوة على كونها لغة « صديقة » ألفاظها تشبه ألفاظ اللغة العادية التى يتحدثها الناس « بالإنجليزية طبعاً » ! . والاسم « لوجو » ليس اختصاراً لعبارة ما كما هو معروف عن لغات الكومبيوتر الأخرى ولكنه مشتق من كلمة يونانية تعنى « الكلمة » أو « الفكرة » !

وقد قام بتصميم لغة لوجو فريق من الباحثين وقام بتطويرها إلى صورتها النهائية التى هى بها الآن الباحث « سيمور بابرت » (S.Papert) فى معامل الذكاء الصناعى بولاية « ماساتشوستس » .

ولغة لوجو لها ارتباط وثيق بلغة LISP التى صممت خصيصاً لأبحاث الذكاء الصناعى فهى تعتبر تطويراً للغة LISP بغرض تسهيل مهمة الأطفال فى تعلم الكومبيوتر وبرمجته فى سن مبكرة جداً ..

لذلك تعتبر لغة لوجو هى اللغة المناسبة لجذب الأطفال إلى عالم الكومبيوتر بدءاً من سن المدرسة . ومع ذلك فسهولة اللغة لا تمنع من كونها لغة قوية تفيد الجميع صغاراً وكبارا . وسهولة اللغة لا تعنى أكثر من أن جهداً كبيراً قد بذله واضعوا اللغة حتى يجعلوا منها لغة سهلة .

ومن الاستخدامات الهامة للغة الرسم بكفاءة عالية ، وخدمة أغراض الذكاء الصناعى والتحكم فى الروبوط فضلاً عن أغراض البرمجة العامة .

وتنفرد لغة لوجو باحتوائها على منشآت للتحكم (control structures) التى تجعلها قادرة على معالجة البيانات فى قوائم (lists) . كما أنها تتميز بالبرامج التى تستدعى نفسها بنفسها وهى خاصية لا تتوفر لكل اللغات الأخرى .

● طرازات لوجو :

توفرت لغة لوجو للعديد من أجهزة الكومبيوتر نذكر منها :

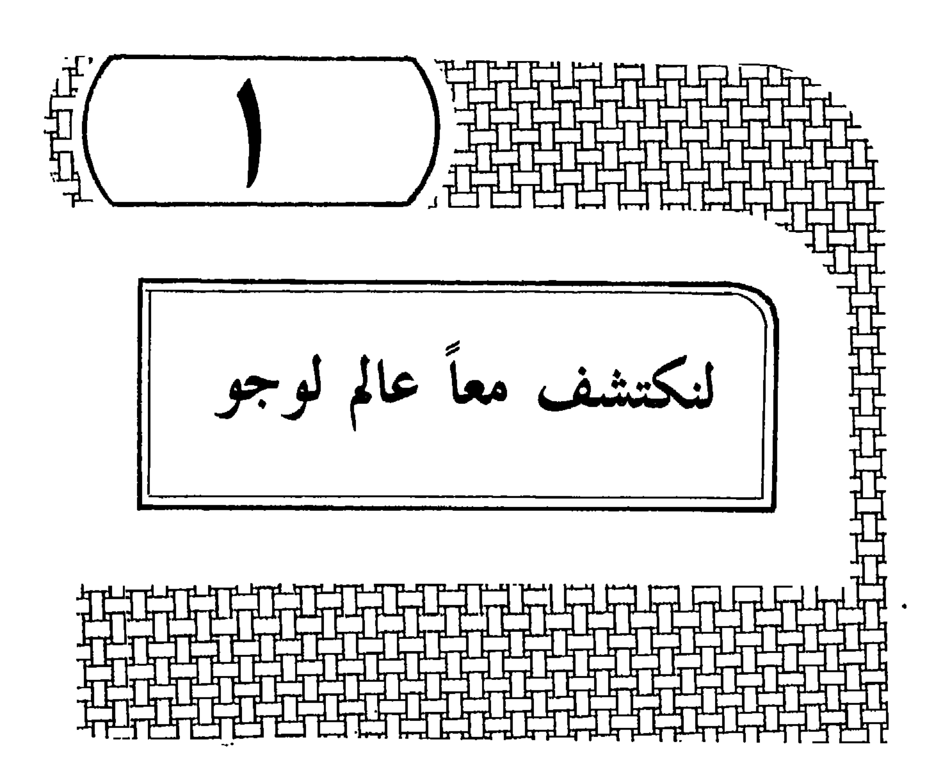
١ ــ الكومبيوتر apple II
٢ ــ الكومبيوتر IBM والأجهزة المتوافقة معه .
٣ ــ الكومبيوتر Radioshake
٤ ــ الكومبيوتر كومودور ٦٤
٥ ــ الكومبيوتر تكساس المنزلى TI-99/4A
٦ ــ الكومبيوتر سنكلير Zxspectrum
٧ ــ الكومبيوتر صخر MSX

وقد قامت شركات البرامج بتطبيق لغة لوجو الآن على معظم أجهزة الكومبيوتر . ولا توجد فوارق كبيرة بين اللهجات المختلفة للغة ، وهى تنحصر أساساً فى استخدام أجهزة الطباعة والقرص المغنطيسى أو الكاسيت ، لذلك فهى لغة قياسية . وبحسب الجهاز يمكنك أن تحصل على لغة لوجو مسجلة على شريط كاسيت أو كارتريدج أو على قرص مغنطيسى .

أقدم هذا الكتاب عن البرمجة بلغة لوجو آملاً أن يكون إضافة حقيقية لمكتبة الكومبيوتر العربية وأن يستمتع به القراء صغاراً وكبارا ..

والله ولى التوفيق ..

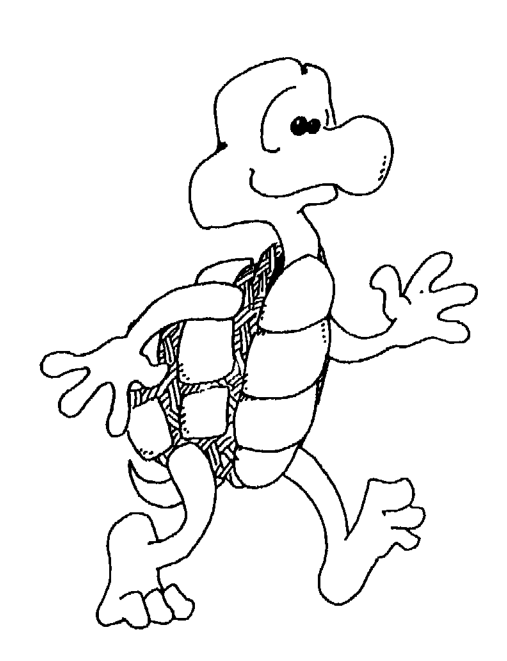
أسامة الحسينى


١
لنكتشف معاً عالم لوجو

● لنتعرف بعائلة لوجو :

إن لغة لوجو لغة حيّة مثيرة . وسوف نتحدث عنها من خلال بعض الأصدقاء الذين يساعدوننا على شرح بعض جوانب اللغة . وبعض الأصدقاء موجود باللغة نفسها مثل السلحفاة البحرية (Turtle) التى تساعدنا فى الرسم بالكومبيوتر ، أما الشخصيات الأخرى فسوف تصاحبنا فقط فى هذا الكتاب فلنتعرف بهم الآن :

[١] السلحفاه :



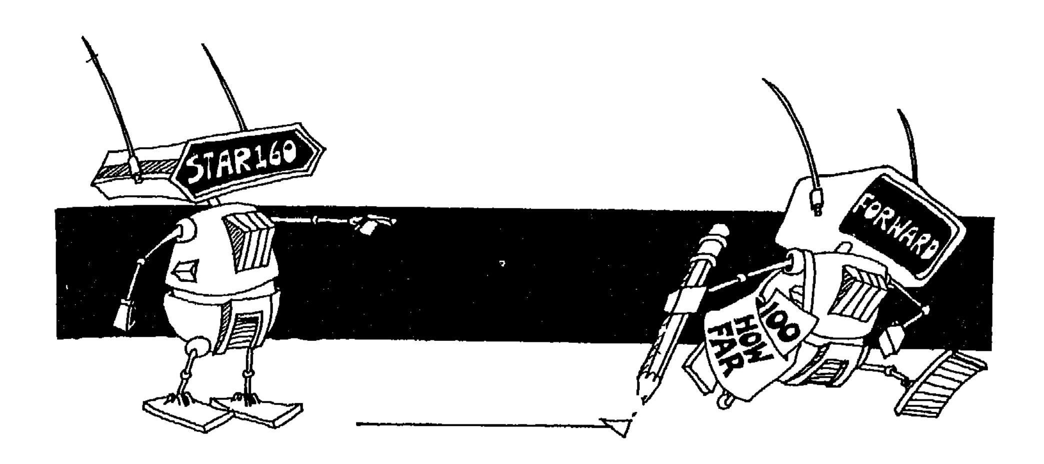
هذه هى الشخصية الرئيسية فى لغة لوجو وسوف تتعامل معها كثيراً عند الرسم على الشاشة ، فهى التى تتلقى منك الأوامر وتقوم بتنفيذها أمامك خطوة بخطوة . ومع بعض أجهزة الكومبيوتر يمكن أن تكون السلحفاه عبارة عن روبوط حقيقى .

٢ الساحر لوجو :



الساحر « لوجو » يختفى بداخل الكومبيوتر ولن نراه على الشاشة للأسف . فهو مشغول طوال الوقت لأنه مسئول عن تلقى جميع الأوامر ومتابعة تنفيذها بواسطة الروبوطات (Robots) التى تساعده فى تنفيذ الأوامر . وسوف نلتقى بالساحر لوجو أحياناً على صفحات الكتاب ليشرح لنا بعض أسرار عمله « عما يحدث بداخل الكومبيوتر » ...

٣ الربوطات المساعدة :

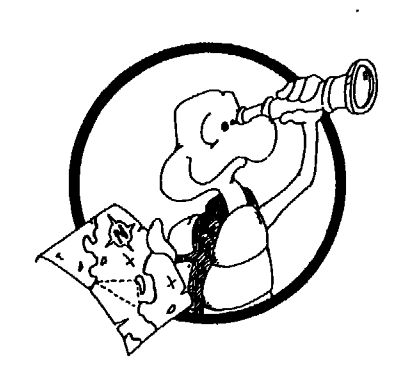


أما الروبوطات المساعدة .. فهى تمثل أوامر لغة لوجو التى يقوم كل منها بأداء عمل محدد يختص به . وهناك نوعان من الروبوطات النوع الأول ذو الشاشة المربعة مثل الروبوط FORWARD الموضح بالرسم وهو يمثل أمراً من أوامر اللغة المعروفة ..أما الروبوط الآخر ذو الشاشة المسدسة STAR160 فهو روبوط مبتكر ! وهو يمثل الأوامر الجديدة التى يمكنك أن تعلّمها للكومبيوتر كما سنعرف عند تقدمنا فى أبواب الكتاب .

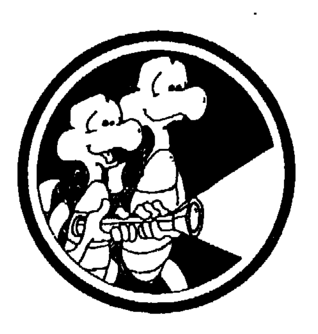
وسواء كانت الروبوطات أصلية أو مبتكرة ، فسوف نلتقى معها عندما يقوم الساحر بشرح « ما يحدث بداخل الكومبيوتر » .

٤ أصدقاء لوجو :

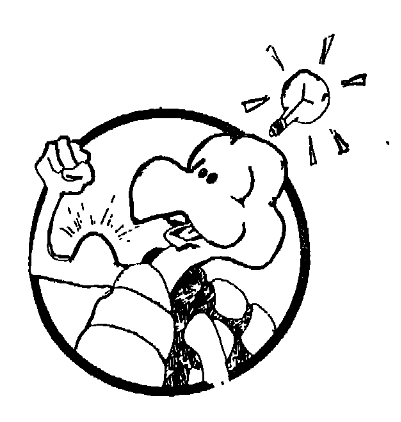
هؤلاء الأصدقاء سوف يتجولون معنا فى رحلتنا داخل هذا الكتاب . أما الصديق الأول فهو يمسك بنظارة معظمّة ويظهر لنا عند التمارين والتجارب التى نقدمها والتى تتطلب منك قدراً من التفكير أو الاستكشاف باستخدام الكومبيوتر .



الصديق الثانى ... هما اثنان متلازمان فى الحقيقة أحدهما يمسك بكشاف لينير الطريق أمامنا . ويظهر هذان الصديقان عند النقاط التى يجب علينا قراءتها بعناية بالغة حيث تكون نقاطاً حاكمة أو ملاحظات هامة .



أما الصديق الثالث فهو يصادفنا عندما نلتقى بفكرة جديدة تتطلب البحث والتقصّى من الأصدقاء هواة البحث والتقصّى .



وبصفة عامة فإن التجارب والتمارين التى يعز عليك تنفيذها سوف تجدها محلولة إما فى آخر الكتاب أو مع موضوعات الكتاب نفسه عندما تتقدم فى القراءة فصلاً بعد آخر .

● إعداد الكومبيوتر للعمل :

أياً كان الكومبيوتر الذى تعمل عليه فإنك تحتاج دائماً إلى « مترجم لغة لوجو » الذى سيجعل الكومبيوتر يفهم أوامر لوجو التى يتلقها . معنى هذا أنك تشترى لغة « لوجو » منفصلة عن الكومبيوتر ، بخلاف « بيسك » . فكل أجهزة الكومبيوتر الصغيرة تستطيع أن تفهم لغة « بيسك » من تلقاء نفسها . وبمجرد أن تضغط على زر التشغيل يمكنك أن تبدأ فى إعطاء الأوامر بلغة بيسك .

أما لغة لوجو فهى عادة تكون مخزنة على شريط كاسيت أو قرص مغنطيسى أو « كارتريدج » .

فإذا كانت اللغة على قرص أو كارتريدج فالأمر لا يتطلب منك سوى وضع القرص أو الكارتريدج فى المكان المخصص له فى الكومبيوتر والضغط على زر التشغيل فيبدأ الكومبيوتر التعامل بلغة لوجو .

● فالجهاز تكساس TI-99/4A مثلاً يحقق لك استخدام لغة لوجو من خلال كارتريدج TI LOGO II .

● كذلك الجهاز صخر MSX يمدّك بلغة لوجو على الكارتريدج « صخر لوجو » الذى يجعل الكومبيوتر يتحدث لوجو باللغتين العربية والإنجليزية .

● أما مع الكومبيوتر سنكلير (Zx spectrum) فيمكنك شراء لغة لوجو مسجلة على شريط كاسيت .

● ومع الجهاز IBM والأجهزة المماثلة له فعادة تكون لغة لوجو مخزنة على قرص مغنطيسى (علاوة على بعض الإمكانات الإضافية على قرص منفصل هو (LWIL) .

وفى حالة ما إذا كانت اللغة مسجلة على شريط كاسيت فإنك تقوم بنقلها (تحميلها) إلى ذاكرة الكومبيوتر تماماً مثل برامج الألعاب العادية ؛ باستخدام الأمر LOAD غالباً . وهذا لا يمنع نقلها إلى القرص المغنطيسى لو توّفر لديك .

وفى جميع الأحوال عندما يبدأ الكومبيوتر العمل بلغة لوجو سوف تشاهد رسالة ترحيب على الشاشة مثل :

WELCOME TO LOGO

عندئذ يمكنك أن تبدأ .

● لنكتب أمراً بلغة لوجو ...

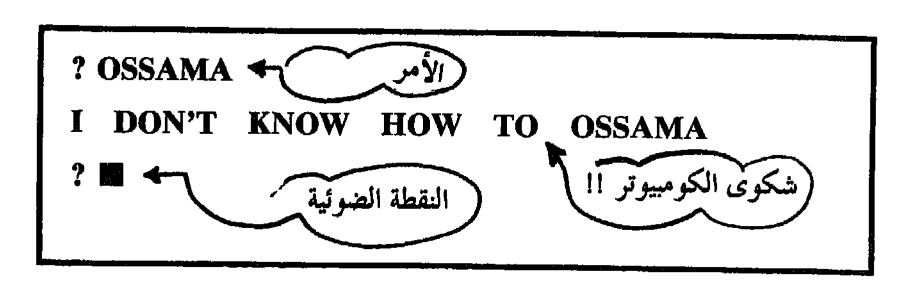
عندما يكون الكومبيوتر مستعداً لتلقى أوامر لوجو سوف تشاهد علامة الاستعداد (؟) على يسار الجزء العلوى من الشاشة يليها النقطة الضوئية (cursor) .

ولتبدأ بكتابة اسمك ولنشاهد معاً ما يحدث :

? OSSAMA ■

النقطة الضوئية   علامة الاستعداد

بعد الانتهاء من كتابة الاسم "OSSAMA" سوف تكون النقطة الضوئية تخفق على يمين الإسم . فإذا ضغطت على الزر (ENTER) فإن الأمر المكتوب يدخل إلى الكومبيوتر وترى الرسالة الآتية على الشاشة :

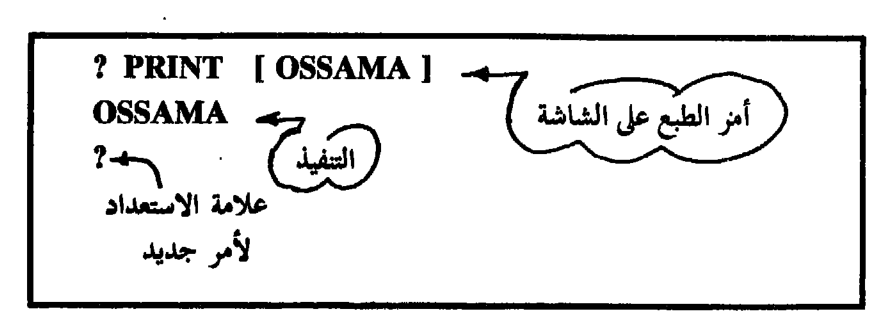


إن الكومبيوتر يشتكى لأنه لا يعرف كيف ينفذ هذاالأمر الغريب .. لأن كل ما يكتب على يمين علامة الاستعداد (?) فى الحقيقة يمثل أمراً للكومبيوتر (COMMAND) .

فإذا أردت إعطاءه أمراً لكتابة اسمك فلنستخدم الأمر PRINT بالصورة الآتية :

```
? PRINT [ OSSAMA ]
```

عندئذ يظهر الاسم على الشاشة مطبوعاً تحت الأمر مباشرة كالآتى :



إذن فقد تعلمنا **أول أمر بلغة لوجو** وهو أمر الطباعة PRINT ولا ننسى استخدام الأقواس المربعة [ ] لنضع بينهما ما نريد طباعته على الشاشة .

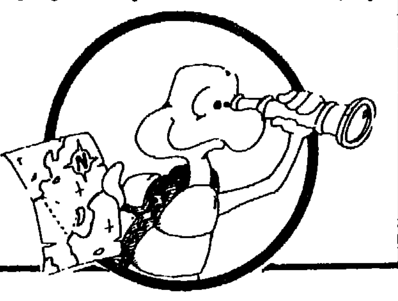
تجـربة

*** عرفنا الآن أن إعطاء الأمر للكومبيوتر بلغة لوجو له صيغة معينة . فإذا خرجنا عنها فإنه يشكو ، وعادة نسمى الشكوى « رسالة خطأ » (error message) .**

والكومبيوتر كثير الشكوى عندما لا يفهم ما نريده أن يفعله . فلتجرب الآن بعض هذه الأوامر ذات الصور والأشكال المتعددة ، لكى تعرف أى الأوامر ستجعله يشكو وأيها سوف ينفذه .

وهذا تدريب جيد لكى تتعرف على بعض رسائل الأخطاء التى يوجهها إليك الكومبيوتر بلغة لوجو :

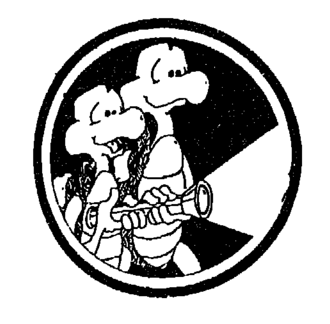
```
PRINT
PRINT   OSSAMA
PRINTOSSAMA
PRINT [ OSSAMA ]
```





الساحر لوجو .. يشكو من عدم فهمه للأمر !

فلاش

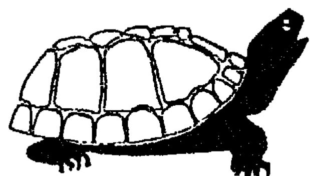


هل لاحظت أننا نستخدم الحروف الكبيرة فى لغة لوجو ؟ .. إنها قاعدة قياسية أن تُكتب لغة لوجو بالحروف الكبيرة ومع ذلك فسوف نصادف بعض الأوامر مكتوبة بالحروف الصغيرة فبعض أجهزة الكومبيوتر تتسامح فى هذه النقطة . لكنها عادة حميدة أن تلتزم بالقاعدة حتى تتقبل أوامرك كل أجهزة الكومبيوتر .

● لا تخش أن تتلف الكومبيوتر :

إنه شيء مفيد حقاً عندما تشرع فى تعلم لغة مثل **لوجو** ــ وهى تتميز بالتنفيذ الفورى للأوامر المعطاه لها ــ أن تجرب إعطاء ما تشاء من الأوامر وترتكب ما شئت من الأخطاء . فبذلك يمكنك أن تتعرف على حدود الكومبيوتر فى لغة **لوجو**وتتعرف على الأخطاء التى يتقبلها والأخطاء التى يشكو منها .

ولا تخش أن تعطى الكومبيوتر أمراً ما . فلن يتلف من الأوامر . وجرّب واكتشف .



● **هيا نلتقى بالسلحفاه :**

والآن نحن على استعداد لمقابلة السلحفاه التى سوف تساعدنا على الرسم بالكومبيوتر .

ولنكتب الآن الأمر الآتى ثم نضغط على الزر ENTER :

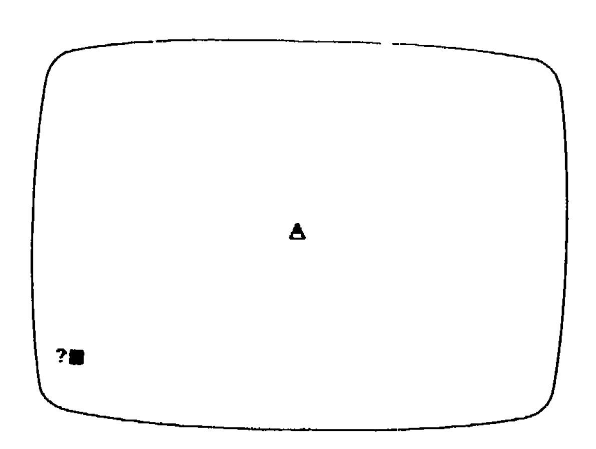
CS

هل شاهدت النتيجة ؟

إنه أمر تنظيف الشاشة وهو اختصار للكلمة الآتية :

CEEARSCREEN

ويمكنك استخدام الكلمة كاملة أو الاختصار CS أيهما تحب . ففى الحالتن سوف تحصل على شاشة نظيفة تماماً من الكتابة السابقة وسوف ترى السلحفاه البحرّية فى وسط الشاشة وعلامة الاستعداد تنتظرك فى الركن الأيسر السفلى هذه المرّة .

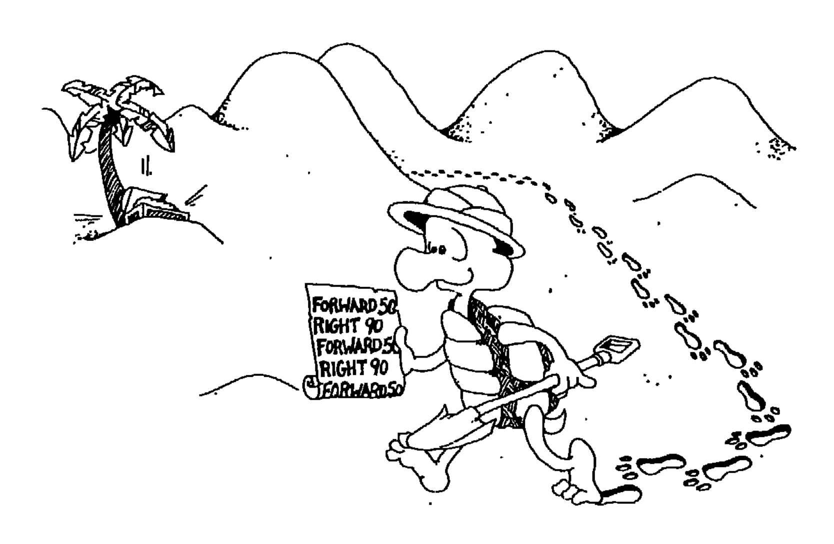


السلحفاه فى منتصف الشاشة بعد تنفيذ الأمر CS

والسلحفاه كما ذكرنا يمكن استبدالها بروبوط حقيقى مع بعض طرازات الكومبيوتر . لكنها مع ذلك وهى على شاشة الكومبيوتر يمكن توجيهها لتتحرك

فى أى اتجاه باستخدام أوامر مفهومة واضحة المعنى مثل « إلى الأمام .. عشر خطوات » أو « إلى الخلف خمس خطوات » أو « استدر لليمين أو لليسار » .

وبهذا فإن السلحفاه بالنسبة لمبرمج لوجو هى هذا الرحّال الذى يحمل خريطة عليها إرشادات تدله على الطريق وسط الصحراء القاحلة .



السلحفاه تحمل الخريطة كمرشد للطريق فى الصحراء

اكتب الآن الأمر الآتى ثم اضغط على الزر ENTER (الزر ذو العلامة ↵ فى الجهاز IBM) :



ماذا حدث على الشاشة ؟

لقد تحركت السلحفاه إلى الأمام بمقدار ٥٠ خطوة .

أعط الآن الأمر الآتى يليه الضغط على الزر ENTER :



تتحرك السلحفاه للخلف مائة خطوة حتى تصبح فى الجزء السفلى من الشاشة .

الآن يمكننا تحريك هذه السلحفاه فى الاتجاهات الأربعة باستخدام أربعة أوامر هى :

الأمـر	اختصار الأمر	المعنى
FORWARD	FD	للأمام
BACK	BK	للخلف
LEFT	LT	لليسار
RIGHT	RT	لليمين

وسوف نلاحظ دائماً أن لغة **لوجو** تتميز بأنها تمنحنا الاختيار بين استخدام كلمات ذات معنى مثل LEFT و RIGHT أو كلمات مختصرة مثل LT ، RT مكافئة لها تماماً . وتتميز الكلمات الكاملة بأنها لا يمكن أن تُنسى لا سيما إذا كنّا نعرف معناها أما الاختصارات فهى توفّر الوقت .

وهذه الأوامر الأربعة تحتاج معها لبعض الأرقام حتى يكتمل معنى الأمر . فلنجرب هذه المجموعة من لأوامر المتتالية :

اكتب هذه الأوامر واحداً بعد الآخر وأدخل كل أمر بالضغط على الزر ENTER :

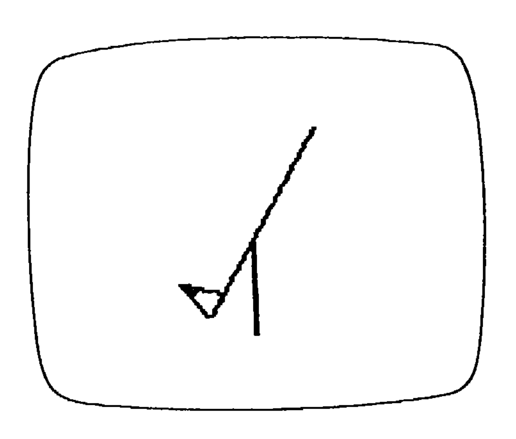


أو أدخل الأوامر مختصرة بهذه الصورة :



ماذا حصلت عليه بعد تنفيذ هذه الأوامر ؟

لو أنك كتبت هذه الأوامر بصيغتها السليمة (سواء العادية أو المختصرة) فإنك سوف تحصل فى النهاية على الشكل الآتى :



نتيجة تنفيذ مجموعة أوامر الحركة للسلحفاه

فلنتفهم معاً نتيجة كل أمر على حدة :

الأمر الأول FD 40 معناه « تحرك إلى الأمام بمقدار ٤٠ خطوة » .



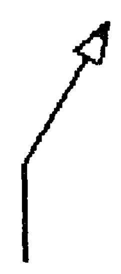
FORWARD 40

الأمر الثانى RT 30 معناه « استدر إلى اليمين بزاوية قدرها ٣٠ درجة » .



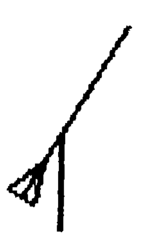
RIGHT 30

الأمر الثالث FD 50 معناه « تحرك إلى الأمام بمقدار ٥٠ خطوة » .. وبالطبع ستكون الحركة فى الاتجاه الجديد الذى يحدده السهم .



FORWARD-50

الأمر الرابع BK 80 معناه « تحرك عكس اتجاه السهم أو إلى الخلف بمقدار ٨٠ خطوة » .



BACK 80

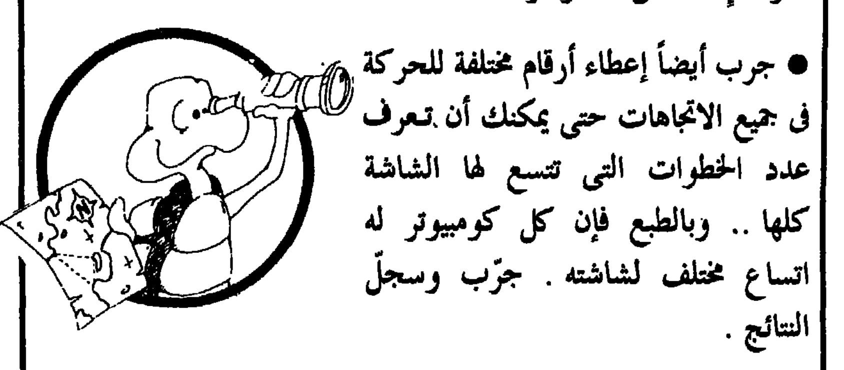
أما الأخير فمعناه تغيير الاتجاه ولكن إلى اليسار (LEFT)بمقدار ٩٠ درجة ، وينتج عنه الرسم الموضح بالشكل النهائى .

نخرج من هذا بأن الرقم الذى يعقب كل من FD ، BK يمثّل عدد الخطوات التى تتحركها السلحفاه .

أما الأرقام التى تعقب كل من LT ، RT فهى تمثل زاوية دوران السهم يميناً أو يساراً .

تجربـــة

● لو أنك نسيت أن تكتب رقماً بعد أمر الحركة FD فماذا تتوقع أن تكون الإجابة من لغة لوجو ؟



● جرب أيضاً إعطاء أرقام مختلفة للحركة فى جميع الاتجاهات حتى يمكنك أن تعرف عدد الخطوات التى تتسع لها الشاشة كلها .. وبالطبع فإن كل كومبيوتر له اتساع مختلف لشاشته . جرّب وسجلّ النتائج .

٢

التحكم فى السلحفاه

• عالـــم السلحفـاه :

إن عالم السلحفاه مازال متسعاً وغنياً بأوامر الحركة والرسم والتحكم والتلوين .

وقبل أن نتعرف بالمزيد من الأوامر الجديدة دعنا نلخّص ما تعلمناه أولاً :

*** الأمر CLEARSCREEN أو CS**

يمسح ما على الشاشة من كتابة أو رسومات ويضع السلحفاه فى المنتصف تماماً جاهزة لتلقى أوامر جديدة .

*** الأمر FORWARD أو FD**
والأمر BACK أو BK

للحركة للأمام وللخلف ، وكلاهما يحتاج لإدخال رقم يمثل عدد الخطوات ونسميه الرقم المُدخل (INPUT) .

*** الأمر RIGHT أو RT**
والأمر LEFT أو LT

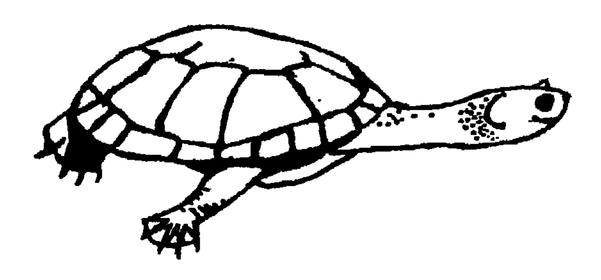
لدوران السلحفاه يميناً أو يساراً وتحتاج أيضاً هذه الأوامر لرقم مُدخل يمثل زاوية الدوران بالدرجات .



وبالإضافة إلى ما سبق ربما تحب أن تكتب أعلى الشاشة وفى هذه الحالة تستخدم الأمر الجديد **TEXTSCREEN** بمعنى شاشة الكتابة ويختصر إلى **TS** .

وفى هذه الحالة تختفى السلحفاه من منتصف الشاشة وترى علامة الاستعداد أعلى يسار الشاشة . وهذه الشاشة تصلح لأداء العمليات الحسابية أو لكتابة برنامج لوجو كما سنعرف فى الأجزاء المقبلة .

ومع ذلك فعند إعطاء أى أمر من أوامر الحركة التى عرضناها فإن الكومبيوتر يعود تلقائياً إلى شاشة السلحفاه .



● **المزيد من الأوامر للسلحفاه PENUP, PENDOWN :**

قد نرغب أحياناً فى رفع القلم عن صفحة الرسم ونقله إلى موضع ما وبدء الرسم من هذا الموضع الجديد .

وكل ما نقوله فى لغة **لوجو** للسلحفاه هو الأمر « **ارفع القلم** » أو **بالإنجليزية PENUP** فإذا تلقت السلحفاه هذا الأمر فإنها تتحرك وفقاً للأوامر التى تتلقاها ولكنها لا ترسم لأن القلم مرفوع .

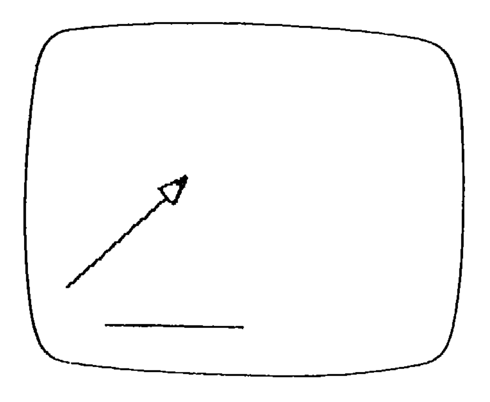
فإذا أردنا أن نبدأ فى الرسم من جديد فإننا نعطيها الأمر « اخفض القلم » أو PENDOWN عندئذ تبدأ السلحفاه فى الرسم من جديد وفقاً للأوامر التى تتلقاها . والأمر **PENUP** يمكن اختصاره إلى **PU** . والأمر **PENDOWN** يمكن اختصاره إلى **PD** .

هل نبدأ فى الرسم من جديد باستخدام هذين الأمرين ؟

لنعط الكومبيوتر أولاً الأمر CS لتنظيف الشاشة ثم تبدأ فى تنفيذ مجموعة الأوامر التالية :

```
CLEARSCREEN
PENUP
FORWARD 50
RIGHT 90
PENDOWN
FORWARD 50
PENUP
RIGHT 45
FORWARD 20
PENDOWN
RIGHT 90
FORWARD 50
```

والشكل الآتى بعد يوضح نتيجة تنفيذ هذه المجموعة من الأوامر :

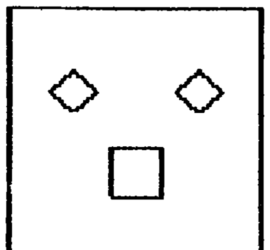


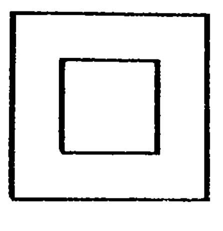
نتيجة تنفيذ مجموعة الأوامر السابقة باستخدام PD , PU

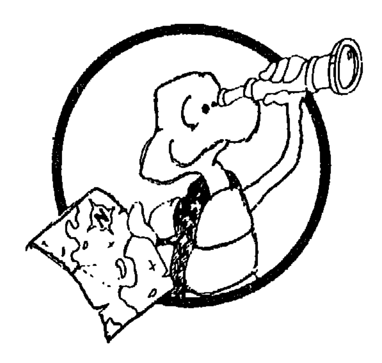
تمرين

هل تستطيع الآن أن ترسم الأشكال الموضحة بعد فى الشكل باستخدام ما تعلمناه من أوامر .

لا بأس أن تحاول الآن فإذا لم توّفق فسوف نعرف معاً كيفية تنفيذ مثل هذه الرسومات فى الفقرات المتقدمة من الكتاب .







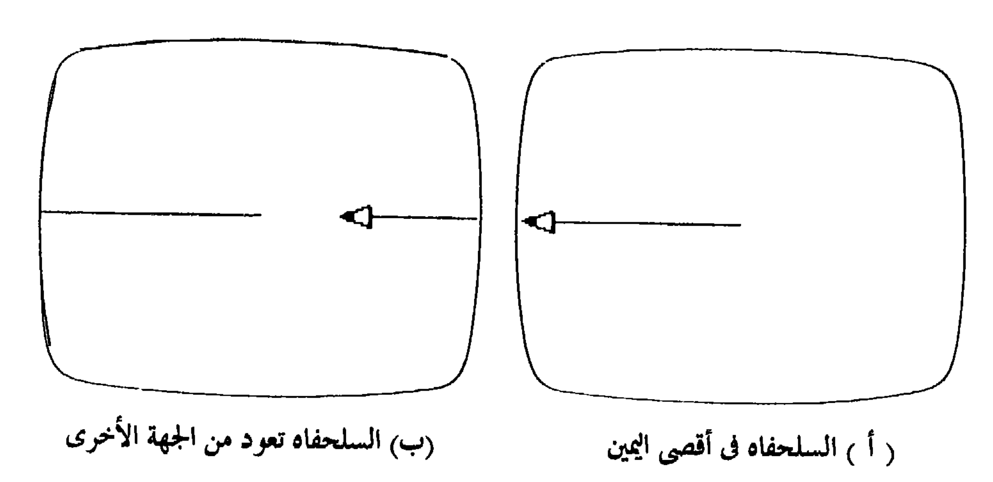
• السلحفاه تحُزِم الشاشة :

بعد أن تتعرف على عدد الخطوات التى يمكن أن تحتويها الشاشة فى الاتجاهين الأفقى والرأسى . يمكنك أن تعرف **إحداثيات** أى نقطة على الشاشة . وعدد الخطوات يختلف من جهاز إلى آخر .

ولكن من التجارب الطريفة التى يمكنك أن تجربها إعطاء الأمر FD مع تجربة بعض الأرقام التى تعتقد أنها تعبّر عن بعد النقطة التى تقع أعلى الشاشة . ومرّة بعد مرة سوف تعثر على الرقم المطلوب .

وكرر الأمر أيضاً بالنسبة ليمين ويسار وأركان الشاشة .

ومن الخصائص المميزة للسلحفاه أنها لن تشكو إذا أعطيتها أرقاماً أكبر من حدود الشاشة فهى تعود دائماً من الجهة الأخرى كما فى الشكل التالى :

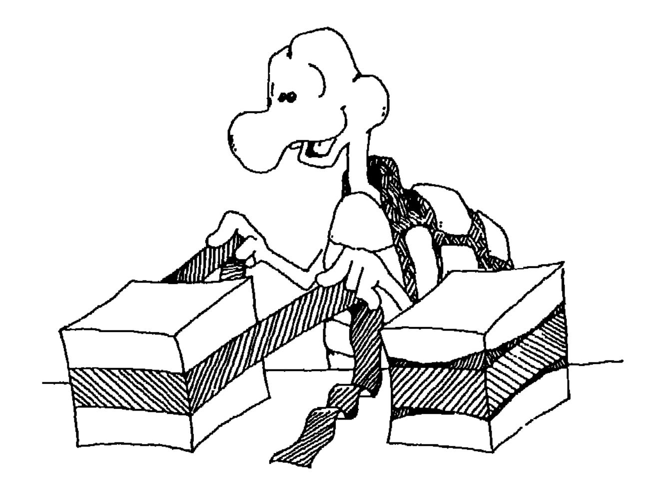


(أ) السلحفاه فى أقصى اليمين

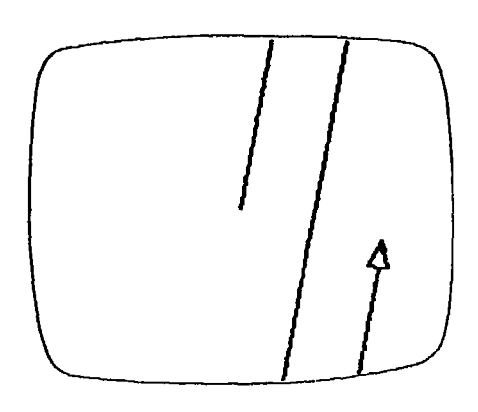
(ب) السلحفاه تعود من الجهة الأخرى

بل يمكن استخدام هذه الخاصية فى تحزيم الشاشة تماماً كما تقوم بتحزيم علبة هدية بشريط من « السوليفان » .

وهذا يتم بإدارة السلحفاه قليلاً وإعطاء الأمر FD مع رقم كبير جداً .



السلحفاه تقوم بتحزيم الهدايا



خاصية التحزيم wrapping

● لنرسم مربعاً بالسلحفاه :

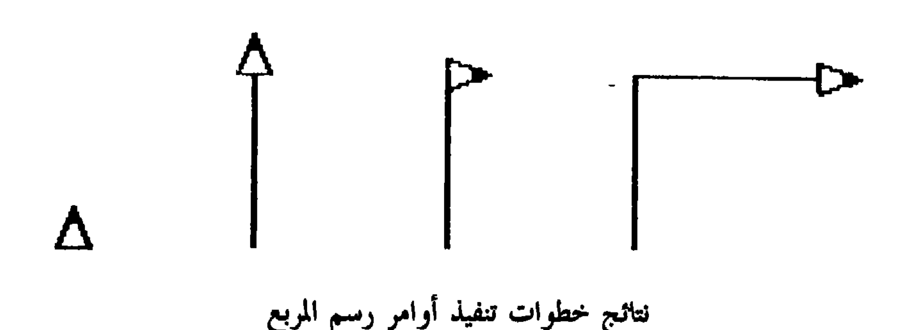
أعط سلسلة الأوامر التالية :

FD 50

RT 90

FD 50

إن هذه الأوامر ترسم جزءاً من مربع ، فالأمر الأول يقول « إلى الأمام خمسين خطوة » ، والثانى يقول « استدر جهة اليمين ٩٠ » ، والثالث يقول « إلى الأمام خمسين خطوة » . بذلك نحصل على الأشكال التالية واحداً بعد الآخر .



نتائج خطوات تنفيذ أوامر رسم المربع

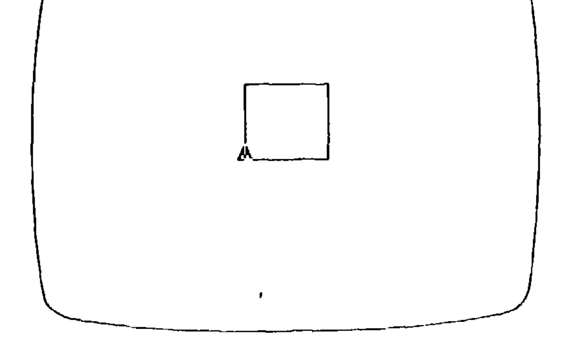
هل تستطيع إكمال الشكل بحيث تُكوّن مربعاً كاملاً ؟

إن كل المطلوب هو إدارة السلحفاه بعد ذلك إلى أسفل وتحريكها نفس عدد الخطوات ثم إدارتها إلى اليسار وإعادتها إلى الموضع الأول بنفس عدد الخطوات .



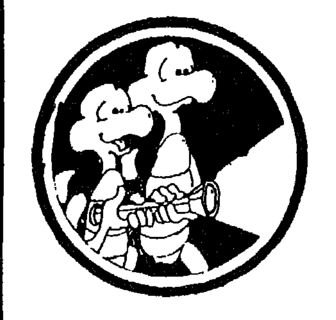
وهذه هى التعليمات كاملة :

```
FD 50
RT 90  ←   إلى اليمين
FD 50
RT 90  ←   إلى أسفل
FD 50
RT 90  ←   إلى اليسار
FD 50
RT 90  ←   إلى أعلى
```



المربع الناتج والسلحفاه
فى وضعها الأصلى

ملاحظة



قد يبدو المربع فى بعض الأحيان مستطيلاً قليلاً وهذا يعتمد على الشاشة المستخدمة فلا تقلق . كما يحدث ذلك أحياناً مع جهاز الطباعة وسوف نلاحظ هذا مع بعض الأمثلة :.

تجربة

حاول باستخدام نفس المنطق أن ترسم مستطيلات بأبعاد مختلفة ، اخترها كما يروق لك .



● إخفاء السلحفاه .. وإظهارها

HIDETURTLE,SHOWTURTLE

بقى شيء هام ينقص المربع الذى قمنا برسمه الآن ، ولا شك أنك لاحظته أيضاً عند محاولتك رسم المستطيلات . هذا هو السلحفاه نفسها التى تبقى دائماً معنا بعد انتهائنا من رسم المربع .

ولا شك أنك ترغب فى إخفائها لمشاهدة المربع كاملاً .

يتم ذلك بإعطاء الأمر HT وهو اختصار الأمر :

HIDETURTLE

بمعنى « أخف السلحفاه » عندئذ تختفى السلحفاه من الرسم وترى المربع « نظيفاً » .

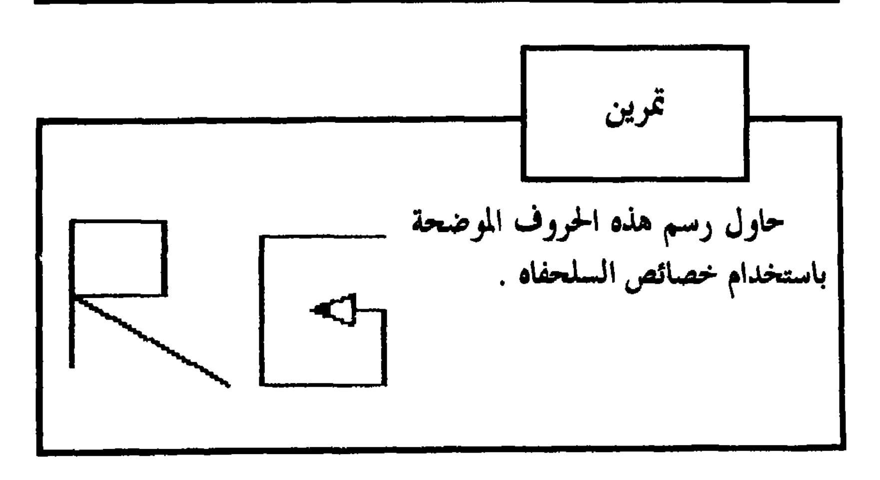
وبالطبع يمكن إعطاء الأمر بأحد صورتيه مختصراً أو كاملاً .

ولإعادة السلحفاه إلى الشاشة يعطى الأمر « أظهر السلحفاه » أى SHOWTURTLE أو بالصورة المختصرة ST . يمكنك الآن إضافة الأمر HT فى نهاية مجموعة الأوامر السابقة وشاهد النتيجة .

وهذا الأمر الجديد سوف يكون نافعاً عند رسم المستطيلات والأشكال الأخرى التى تحاول رسمها .

وهذا ملخص بكلمات لوجو التى تعلمناها مؤخراً :

الأمــر	اختصار الأمر	المعــنى
TEXTSCREEN	TS	إعداد شاشة الكتابة بدون سلحفاه .
PENUP	PU	ارفع القلم .. لا ترسم عند التحرك .
PENDOWN	PD	اخفض القلم .. ارسم عندما تتحرك .
HIDETURTLE	HT	أخف السلحفاه .
SHOWTURTLE	ST	أظهر السلحفاه .



● الرسم بالألوان[1] SETPAL, SETPC

يمكن تكوين الرسومات باستخدام أمرين جديدين هما :

اختيار مجموعة الألوان	SETPAL
(وهو يعنى بالإنجليزية	(set palette
٢ ـــ اختيار لون القلم	SETPC
(وهو يعنى بالإنجليزية	(set pen color

وعادة يتبع الأمر SETPAL إما الرقم 0 أو الرقم 1 .

أما الأمر SETPC فيعقبه أحد الأرقام من 1 إلى 3 .

ومن هذه التوليفة يمكن الحصول على ستة ألوان مختلفة للرسم كالآتى :

● أعط أمر ألوان المجموعة الأولى ← STPAL 1

يمكن مع هذه المجموعة إعطاء الأوامر التالية لكل لون :

(للون الأزرق الفاتح cyan)	SETPC 1
(للون الأرجوانى magenta)	SETPC 2
(للون الأبيض white)	SETPC 3

● أو أعط أمر اختيار ألوان المجموعة الثانية SETPAL 0

ويمكن مع هذه المجموعة إعطاء الأوامر التالية لكل لون :

(للون الأخضر green)	SETPC 1
(للون الأحمر red)	SETPC 2
(للون البنى brown)	SETPC 3

---

(١) تختلف عبارات الألوان باختلاف الأجهزة .

تجربة

جرب الآن إعطاء أمر تلوين أحد المجموعتين (0) أو (1) يليه اختيار لون القلم الذى يروق لك ثم ارسم شكلاً ملّوناً .
(ويمكنك تغيير لون القلم عدة مرات أثناء الرسم) .

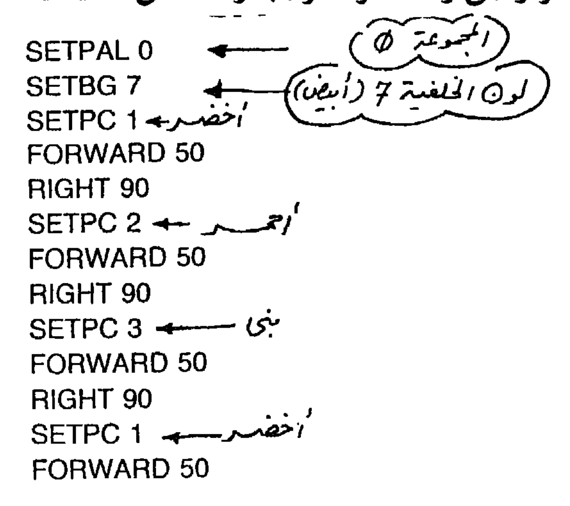
● **تلوين الخلفية** **SETBG**

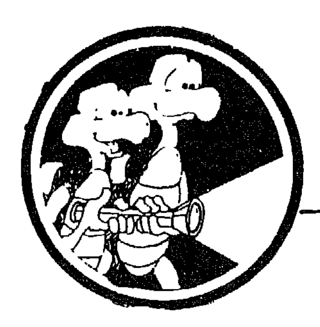
يوجد لون رابع خلاف الألوان 1 ، 2 ، 3 ، بكل مجموعة ألوان ، هذا هو اللون صفر (0) . وهذا اللون يرسم بنفس لون الخلفية ولذلك فهو يمسح الرسم الموجود على الشاشة إذا مرّ فوق .

علاوة على ذلك فإنه يمكن تغيير لون الخلفية باستخدام الأمر SETBG مع إدخال رقم بعدها من (0 إلى 15) وبذلك نحصل على ١٦ لوناً مختلفاً للخلفية .

● **مثـال :**

الآتى بعد مجموعة من الأوامر لرسم مربع ذى أضلاع ملونة بالألوان الأخضر والأحمر والبنى والأخضر بالترتيب وذلك على خلفية بيضاء .





● الألوان .. وأجهزة الكومبيوتر المختلفة :

نظراً لأن خاصية التلوين ترتبط دائماً بنوع الكومبيوتر ، حيث أنها تعتمد على تصميم المعدّة نفسها ، لذلك فإننا يجب أن نشير إلى أن طرق التلوين التى قدمناها هنا تنطبق على أجهزة الكومبيوتر **طراز IBM** والأجهزة المتوافقة معه .

ومع ذلك فإذا كنت تملك جهاز كومبيوتر آخر فالاختلاف ليس كبيراً بين عبارات التلوين . فمع الأجهزة المنزلية غالباً لا تستخدم أمر مجموعات الألوان SETPAL وإنما تستخدم الأمر SETPC لتلوين القلم مباشرة . وكذلك الأمر SETBG لتلوين الخلفية .

والألوان المستخدمة تعتمد على نوع الجهاز فمثلاً **مع الكومبيوتر المنزلى سنكلير** توجد الألوان الآتية (على لوحة الأزرار) :

أزرق	**1**
أحمر	**2**
أرجوانى (magenta)	**3**
أخضر	**4**
أزرق فاتح (cyan)	**5**
أصفر	**6**
أبيض	**7**
أسود	**0**

● مثال على الكومبيوتر سنكلير المنزلى :

```
SETBG  7  ←      خلفية بيضاء
SETPC  4  ←      أخضر
FD  50
RT  90
SETPC  2  ←      أحمر
FD  50
RT  90
SETPC  3  ←      أرجوانى
FD  50
RT  90
SETPC  1  ←      أزرق
FD  50
HT          ←      إخفاء السلحفاه
```

ونلاحظ هنا أن اللون البنى غير موجود بالكومبيوتر سنكلير لذلك استبدلناه باللون الأرجوانى (magenta) رقم 3 .

كما نلاحظ إضافة أمر إخفاء السلحفاه كأمر أخير .

● فلنتقدم خطوة أخرى إلى الأمام :

لا يمانع « لوجو » أن يستقبل مجموعة من الأوامر فى سطر واحد . وفى هذه الحالة لن يتطلب رسم المربع الملون كل هذه السطور المتتالية ، كما أنك لن تضغط على الزر (Enter) إلاّ مرة واحدة . وعلى سبيل المثال يمكنك رسم مربع بمجموعة أوامر فى سطر واحد فقط أو فى سطرين كما يتراءى لك :

```
FD 50 RT 90 FD 50 RT 90
FD 50 RT 90 FD 50
```

● ارسم مربّعاً بأمر واحد فقط REPEAT

لو ألقينا نظرة فاحصة على مجموعة الأوامر المتتالية التى ترسم المربّع لوجدنا أنها عبارة عن تكرار لأمرين متتابعين هما (FD 50) ، (RT 90) .

وتمدّنا لغة لوجو بإمكانية جديدة لتكرار الأوامر وهى الأمر REPEAT بمعنى كرّر وهو يستخدم كالآتى :

? REPEAT 4 [FD 50 RT 90]

(عدد مرات التكرار)

هذا الأمر يؤدى إلى تكرار مجموعة الأوامر التى بين القوسين المربعين ٤ مرات . جرب هذا الأمر سوف تشاهد المربع يرسم على الشاشة ثم تعود السلحفاه إلى وضعها الأصلى .

والأمر REPEAT قد يحتوى بداخل القوسين المربعين على أية أوامر من أوامر لوجو بما فى ذلك الأمر REPEAT نفسه .

● بناء سور حول السلحفاه FENCE, WRAP

رأينا أن السلحفاه يمكن أن تتقبل أمراً للتحرك إلى الأمام مثل :

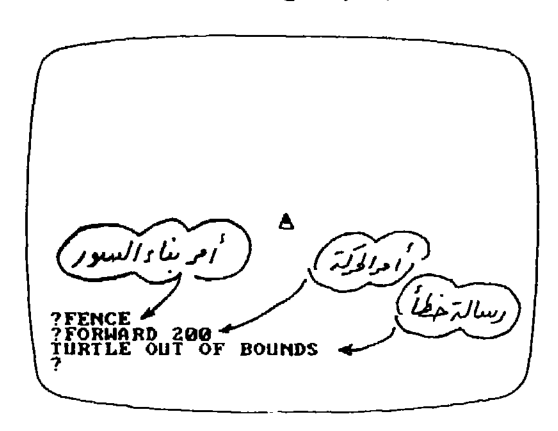
FD 1000

والرقم 1000 بالطبع كبير جداً ويخرج عن حدود أى شاشة . ولكن السلحفاه إزاء هذا الأمر تقوم بعملية التحزيم التى ذكرناها وتعود من الناحية الأخرى للشاشة .

ويمكن إيقاف عملية التحزيم هذه بإعطاء الأمر FENCE بمعنى سور (الحديقة) ، أو السياج الذى يحدّد مساحة من الأرض .

وهذا الأمر يجعل السلحفاه تتقيد بحدود الشاشة فى حركتها ولا تخرج عنها . وفى هذه الحالة . لو أنك أدخلت مع الأمر FD رقماً كبيراً يتعدى حدود

الشاشة فإن السلحفاه لا تتحرك ويرسل لك **لوجو** رسالة خطأ كالآتى :



ويمكن إلغاء هذه الخاصية بإعطاء الأمر WRAP فيعود الحال إلى ما كان عليه ولتجرب هذه الأوامر المتتابعة وشاهد النتائج .

```
FENCE
FORWARD 200
WRAP
FORWARD 200
```

● شاهد السلحفاه من النافذة WINDOW

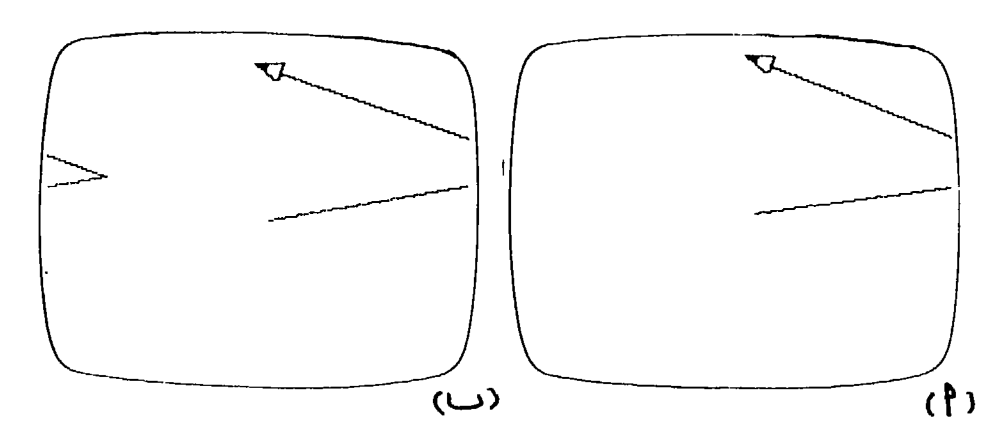
إذا أعطيت الأمر WINDOW فإن السلحفاه يمكنها الحركة خارج حدود الشاشة ، ولكنها متى تركت الشاشة فإنها تختفى . وبذلك تصبح الشاشة كأنها نافذة تحدّد المشهد أمامك .

جرب هذه المجموعة من الأوامر (يمكنك كتابتها مختصرة) :

```
CLEARSCREEN
WINDOW
RIGHT 80
FORWARD 180
LEFT 150
FORWARD 180
```

ونتيجة هذه الأوامر موضحة فى الشكل على شاشة الكومبيوتر IBM.ثم جرب هذه المجموعة من الأوامر وفيها نستبدل الأمر WINDOW بالأمر WRAP الذى يعيد الحال إلى ما كان عليه فتبدأ خاصية التحزيم مرة أخرى . وتظهر النتيجة فى الشكل (ب) .

```
CLEARSCREEN
WRAP
RIGHT 80
FORWARD 180
LEFT 150
FORWARD 180
```



الشكل (أ) باستخدام الأمر WINDOW أما (ب) فباستخدام الأمر WRAP .

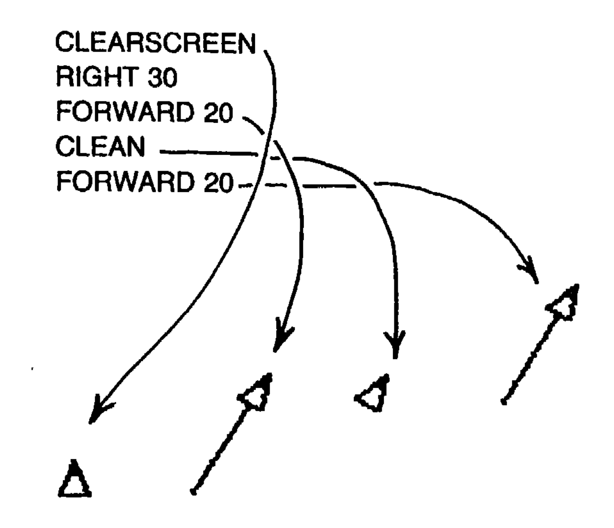
● تنظيف الشاشة بالأمر CLEAN :

هناك فارق أساسى بين تنظيف الشاشة بالأمر (CS) أو (CLEARSCREEN) وبين تنظيفها بالأمر (CLEAN) فهذا الأمر الأخير يمسح الشاشة من الرسم ولكنه يترك السلحفاه فى وضعها الذى كانت عليه .

أما الأمر (CS) فهو يعيد كل شيء إلى أصله بما فى ذلك وضع السلحفاه .

● مثال :

إدخل هذه المجموعة من الأوامر تشاهد النتائج المرسومة بجوارها بالتتابع :



● إعادة السلحفاه إلى بيتها HOME

يمكنك بإعطاء الأمر HOME أن تعيد السلحفاه إلى بيتها الأصلى الذى تبدأ منه الحركة عادة ورأسها متجه إلى أعلى :

● مثال :

أدخل هذه المجموعة من الأوامر وشاهد النتائج :

```
CLEARSCREEN
FORWARD 50
HOME
CLEAN
```

هل لاحظت تأثير الأمر CLEAN الذى جاء بعد الأمر HOME ؟ إن هذه التركيبة من الأمرين CLEAN , HOME تكافىء تماماً الأمر CLEARSCREEN .

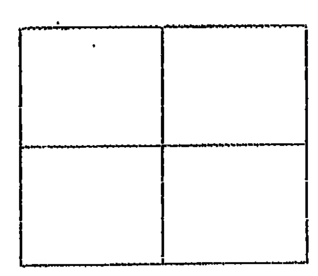
ولكنك لو أدخلت الأمر CLEAN قبل الأمر HOME فإن النتيجة سوف تختلف جرب وشاهد .

```
CLEARSCREEN
FORWARD 50
CLEAN
HOME
```

بالطبع هناك أوامر كثيرة مازالت فى لغة « لوجو » للتحكم فى السلحفاه ولكن بهذا القدر من الأوامر يمكنك أن ترسم ما شئت على الكثير من أجهزة الكومبيوتر .

تجربة

رسمنا من قبل الشكل المربع بأمر واحد باستخدام خاصية التكرار REPEAT . هل تستطيع أن ترسم الشكل الآتى المكوّن من المربّعات باستخدام توليفة معينة من الأمر REPEAT مع أمر رسم المربع ؟ (حاول والإجابة فى نهاية الكتاب) .



● التحكم فى الروبوط STARTROBOT,STOPROBOT

عندما يكون جهاز الكومبيوتر متصلاً بأحد الروبوطات الميكانيكية فإنه يمكن التحكم فيها بأوامر الحركة الخاصة بالسلحفاه هى :

FD , BK , PU , PD , RT , LT

ويتم ذلك بإعطاء الامر :

STARTROBOT

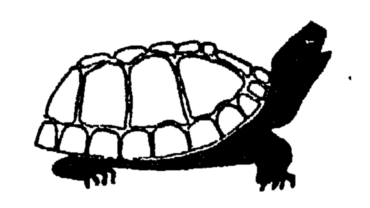
وعند إعطاء هذا الأمر يجب أن يكون قد سبق إعداد برنامج الحركة الذى يتحكم فى الروبوط . فإذا لم يكن البرنامج موجوداً فإن الكومبيوتر يبحث عنه ويحاول تحميله من القرص المغنطيسى أو الشريط الكاسيت .

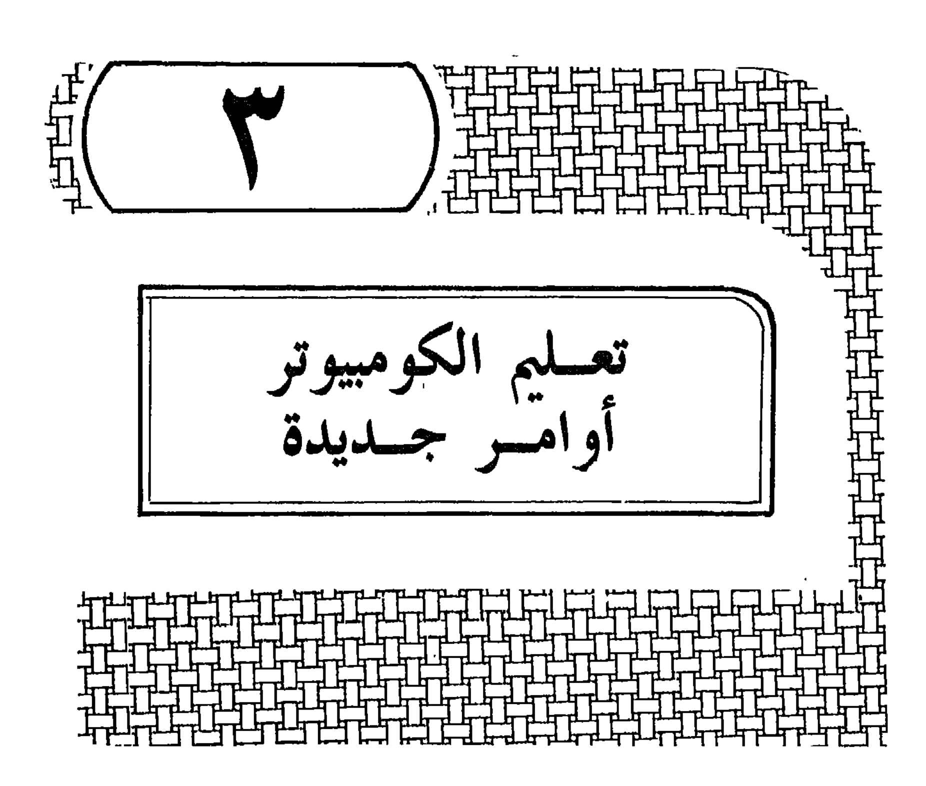
وهذا الأمر قد يختلف من جهاز إلى آخر ولكن عند الحاجة إليه يسهل البحث عن أمر مماثل فى كتالوج استخدام الجهاز بلغة لوجو .

ولإيقاف التحكم فى الروبوط يستخدم الأمر :

STOPROBOT

فيؤدى إلى إنهاء العلاقة مع الروبوط الميكانيكى واقتصار الأوامر على حركة السلحفاه .




٣
تعليم الكومبيوتر
أوامر جديدة

● علّم الكومبيوتر أمراً جديداً TO, END

رغم الاختصارات التى استخدمناها مع أوامر لوجو ، ورغم الوسائل الموفّرة للوقت مثل تكرار الأمر باستخدام أمر التكرار REPEAT ولكن مع هذا كله فإن إدخال عدة أوامر متتابعة للكومبيوتر يصبح عملاً مملّاً بعد فتره من إجادته .

فلو أردنا أن نرسم صندوقاً مربعاً فإننا نضطر فى كل مرة أن نكتب مجموعة من الأوامر مثل :

```
? FD 50     RT 90
? FD 50     RT 90
? FD 50     RT 90
? FD 50     RT 90
```

ألا توجد طريقة أخرى لكى نعطى للكومبيوتر أمراً واحداً مختصراً مثل BOX فيرسم الصندوق المطلوب ؟

ولكننا نعلم أننا لو أعطينا الكومبيوتر أمراً لا يعرفه مثل BOX فسوف يشكو الساحر لوجو قائلاً : « أنا لا أعرف كيف أرسم صندوقاً » وبلغته الإنجليزية الطريقة يقول :

"I don't know how to box" !



السـاحر لوجو .. يفتش كل خزائنه بحثاً عن أمر إسمه BOX
ولكن بلا جدوى ..

ومع ذلك يمكننا تنفيذ ما نريد بطريقة ما .

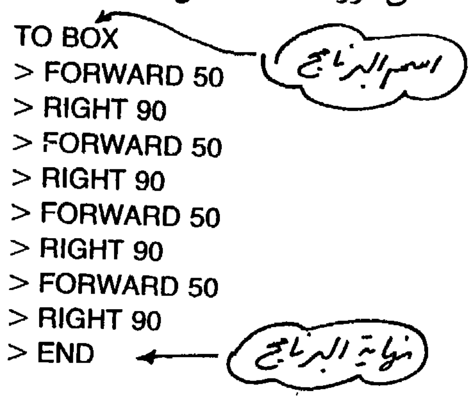
فهناك وسيلة لحفظ مجموعة الأوامر التى ترسم الصندوق فى ذاكرة الكومبيوتر فيما يسمى بالبرنامج الفرعى[1] أو البرنامج (Procedure) .

ويتم ذلك بإعطاء الأمر الآتى (على شاشة الكتابة TEXTSCREEN) :



عند إعطاء هذا الأمر تظهر العلامة (<) عند أول السطر ويتوقف الكومبيوتر عن تنفيذ أوامر لوجو ويتوقف أيضاً عن الاعتراض إذا أنت أخطأت فى الكتابة بلغة لوجو . فهو يمنحك الآن صفحة تكتب فيها ما تشاء . وبعد أن تنتهى من الكتابة سوف يحفظ كل ما كتبته فى الذاكرة تحت الاسم BOX .

وبذلك فإن البرنامج BOX بعد إتمامه . سوف ينضم إلى مجموعة الأوامر التى يفهمها وينفذها الكومبيوتر مثل FD , BK , RT ... إلخ . ولذلك **لا يجوز أن تطلق على برنامجك اسماً من هذه الأسماء** التى يفهمها الكومبيوتر . فهذه الأسماء فى الحقيقة هى برامج أيضاً ولكن سبق تخزينها فى لغة لوجو ، وهى تسمى البرامج الأولية (primitve procedures) . فلنبدأ الآن بكتابة البرنامج BOX ولا تنس أن تضغط على الزر Enter بعد كل سطر .



(١) يستخدم أيضا مصطلح الروتين (routine)

وبعد أن تفرغ من كتابة سلسلة الأوامر التى ترسم الصندوق يجب أن تتبعها بالأمر END . هنا فقط يستيقظ اللوجو من جديد ويعرف أنك انتهيت من الكتابة وأنه جاء دوره ليبدأ فى تنفيذ الأوامر لذلك تظهر علامة الاستفهام مرة أخرى على يسار الشاشة بعد أن يرسل إليك الرسالة الآتية .

BOX DEFINED

بهذه الرسالة الأخيرة يصبح البرنامج BOX أمراً من أوامر لوجو المحفوظة فى ذاكرته ويمكنك تنفيذ جميع خطوات البرنامج بالأمر التالى :

? BOX

بمجرد أن تضغط على الزر Enter تبدأ السلحفاه فى رسم الصندوق المربع بشرط أن يكون البرنامج سليماً وخالياً من الأخطاء . فإذا لم يكن كذلك فإن الكومبيوتر يتوقف عند الخطأ ويرسل لك شكوى بما أعاقه عن العمل .

فما هو التصرف فى حالة وجود خطأ بالبرنامج ؟

الأمر البديهى هو إعطاء الأمر TO BOX مرة أخرى وكتابة البرنامج من جديد . ولكن لا .. هناك حل أفضل بكثير .

● إصلاح برنامج لوجو EDIT

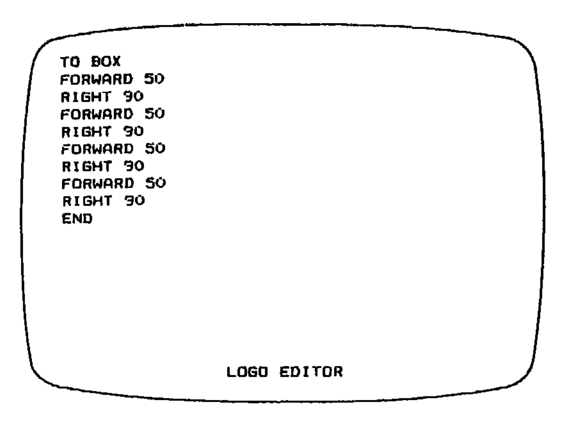
يمكنك استرجاع البرنامج BOX باستخدام وسيلة داخلية ضمن لغة لوجو تسمى « المحرر » (Screen Editor) .

ويستدعى المحرر مُحتوياً على البرنامج BOX بالأمر :

? EDIT "BOX"

بمجرد كتابة هذا الأمر تجد البرنامج BOX معروضاً على الشاشة مسبوقاً بكلمة TO BOX وجاهزاً على الإصلاح .

والمحرر لا يراجع الأخطاء ولا ينفذ الأوامر أيضاً ، فهو مجرد وسيلة للكتابة أو لإصلاح الكتابة السابقة . ويمكنك بداخله أن تتحرك فى جميع الاتجاهات باستخدام الأسهم . كما يمكنك استخدام أزرار المسح (.Del) وحشر الحروف (.ins) وسائر أزرار الكتابة المعروفة .



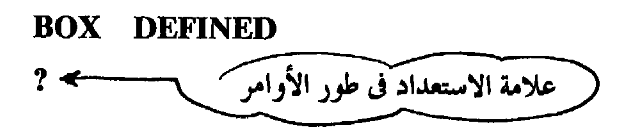
محرر لوجو يعرض البرنامج BOX

ويقال أن الكومبيوتر فى هذه الحالة موجود فى **طور التحرير** **(edit made)** أما الطور العادى الذى عرفناه من قبل فهو **طور الأوامر** **(Command mode)** ويتميز بعلامة الاستفهام التى تظهر على اليسار . وللخروج من طور التحرير إلى طور الأوامر يلزم استخدام زر خاص . ففى الكومبيوتر IBM والطرازات المشابهة نستخدم الزر Esc . وفى الأجهزة الأخرى توجد عادة أزرار مشابهة فى الوظيفة وربما تحمل أسماء أخرى . ففى الكومبيوتر سنكلير مثل يلزم الضغط على عدة أزرار هى :

symbol shift + caps shift + C

وبصفة عامة يتم الرجوع لتعليمات استخدام برنامج لوجو مع الأجهزة الصغيرة . وعند الخروج من المحرر سوف تشاهد الرسالة :

BOX DEFINED

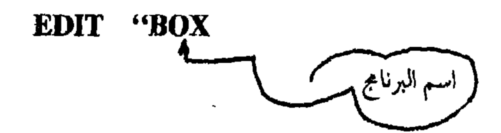


بهذا يكون البرنامج محفوظاً فى ذاكرة الكومبيوتر كأمر من أوامره المعروفة مسبقاً ويمكنك فى هذه الحالة رسم الصندوق باستخدام الأمر البسيط BOX .

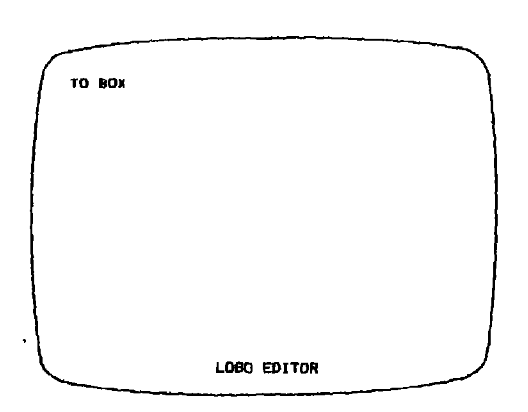
● **كتابة البرنامج باستخدام المحرر**

فضلاً عن إصلاح البرنامج باستخدام المحرر فإنه يمكن من الأصل كتابة البرنامج الجديد باستخدام المحرر . ويتم ذلك باستدعاء المحرر بالأمر :

EDIT "BOX



عندئذ سوف تشاهد العبارة TO BOX مكتوبة أعلى الشاشة أما بقية الشاشة فهى جاهزة على الكتابة .



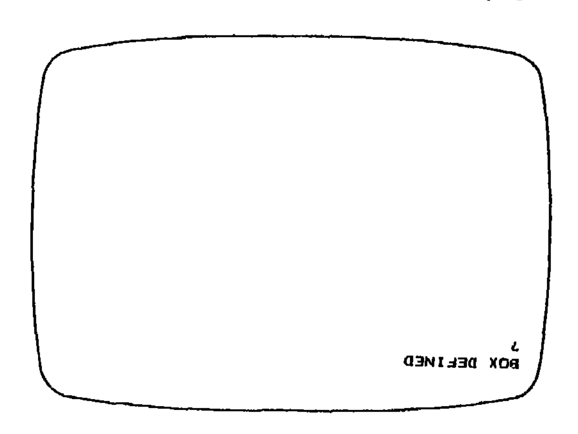
شاشة محرر لوجو جاهزة على الكتابة

بعد ذلك تبدأ فى كتابة البرنامج سطراً بسطر مع الضغط على الزر **Enter** بعد كل سطر (وهذا موضح بالعلامة (↵) المرسومة على زر الكومبيوتر IBM) :

```
FORWARD 50 ↵
RIGHT 90 ↵
FORWARD 50 ↵
RIGHT 90 ↵
FORWARD 50 ↵
RIGHT 90 ↵
FORWARD 50 ↵
RIGHT 90 ↵
END ↵
```

(اضغط Enter)

بعد كتابة البرنامج يتم الخروج من طور التحرير على الزر ESC (أو ما يعادله فى الأجهزة الأخرى) . فتظهر الرسالة BOX DFINED أعلى الشاشة كالآتى :



تم تعريف البرنامج BOX

بهذا يكون البرنامج BOX جاهزاً على التنفيذ

وبالطبع فى حالة حدوث خطأ بالبرنامج يمكن استدعاء البرنامج BOX بنفس الطريقة وإصلاحه .

كما يمكن مسح البرنامج BOX من ذاكرة الكومبيوتر بالأمر :

ERASE "BOX

أو بالصورة المختصرة : **ER "BOX**

ويمكن تخزين أكثر من برنامج فى نفس الوقت فى ذاكرة الكومبيوتر بنفس

الطريقة ، حيث تخصص لغة لوجو مساحة من الذاكرة تسمى حيز العمل (working space) وفى هذا الحيز تخزن البرامج التى يكتبها مستخدم اللغة وهذه خاصية فريدة للغة لوجو .

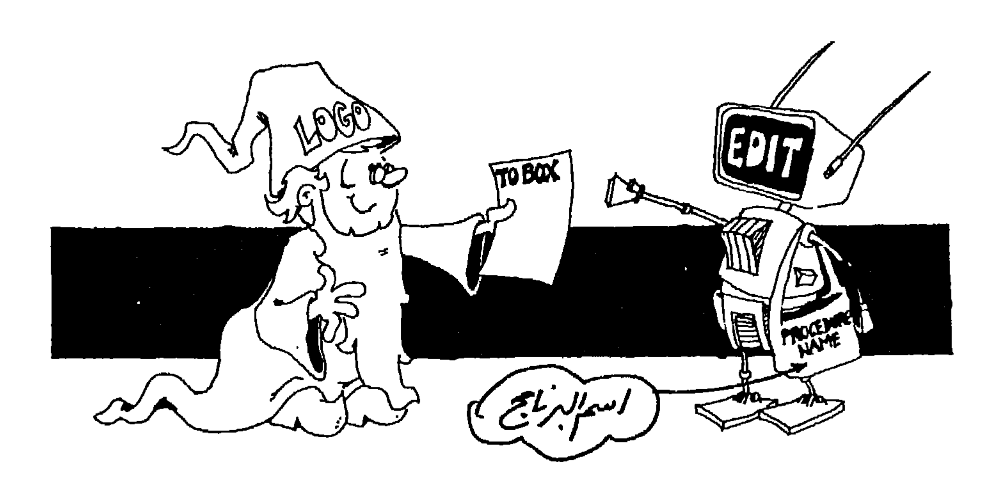
ويمكن مسح جميع البرامج من حيز العمل بالأمر :

ERALL

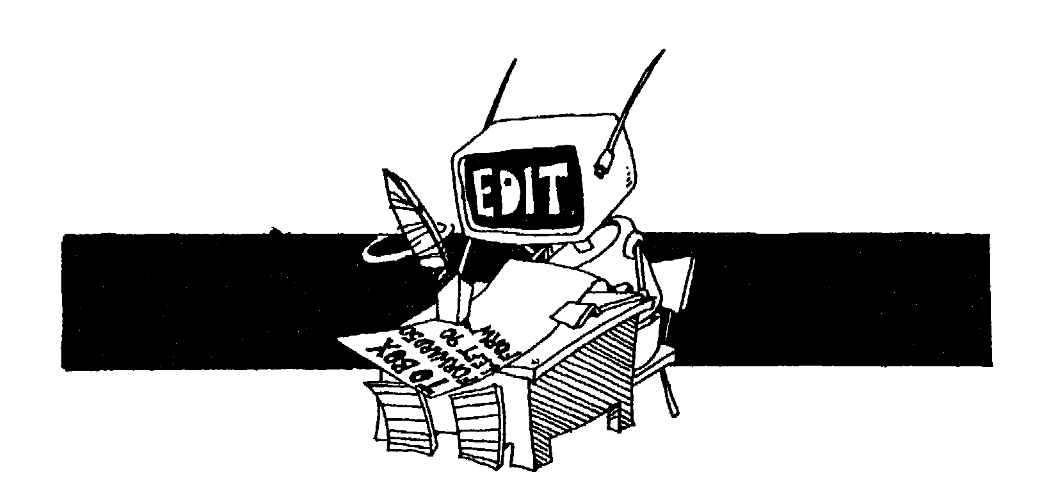
● ماذا يحدث بداخل الكومبيوتر ؟

فيما يلى سوف يساعدنا الساحر لوجو على توضيح بعض التفصيلات التى تحدث بداخل الكومبيوتر عند استخدام المُحَرِّر Editor لكتابة وحفظ البرنامج BOX ثم تنفيذه فى النهاية .

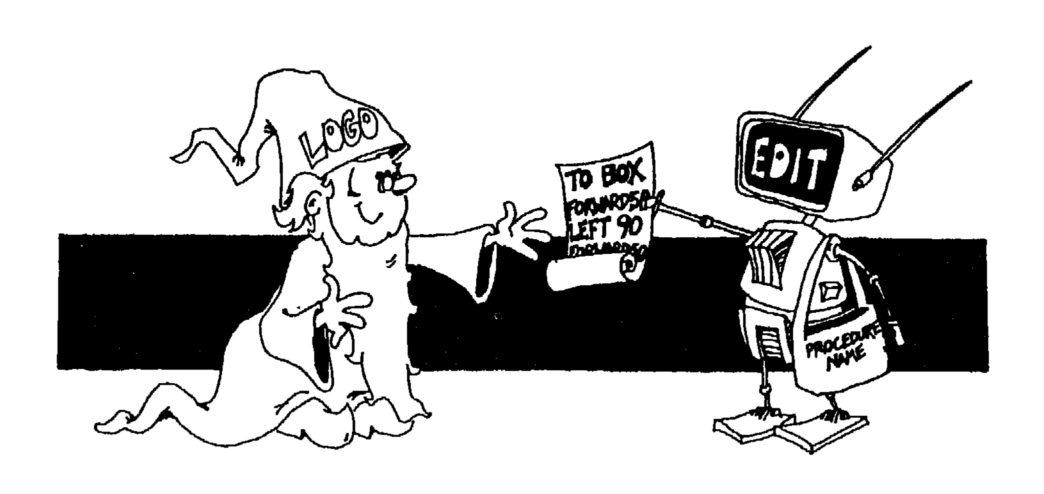
ويعاون الساحر لوجو كل من الروبوط EDIT الذى يمثل المُحَرِّر والروبوط BOX الذى يمثل البرنامج .



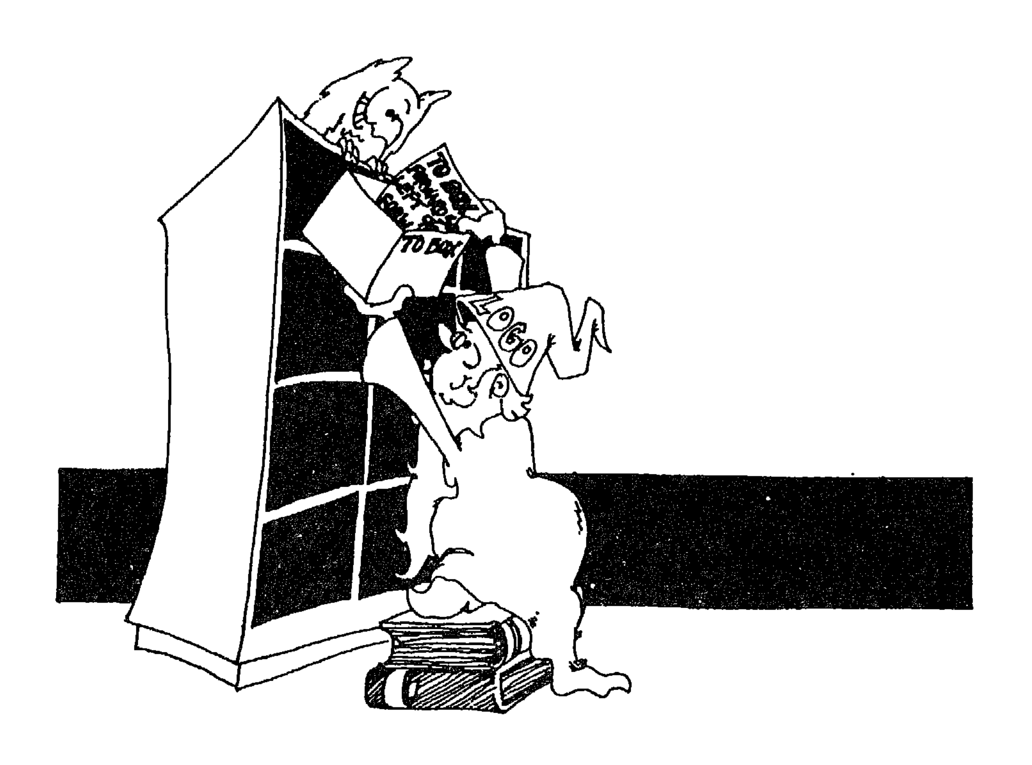
(أ) الساحر لوجو يستدعى الروبوط EDIT ويخبره باسم البرنامج المطلوب إنشاؤه BOX .



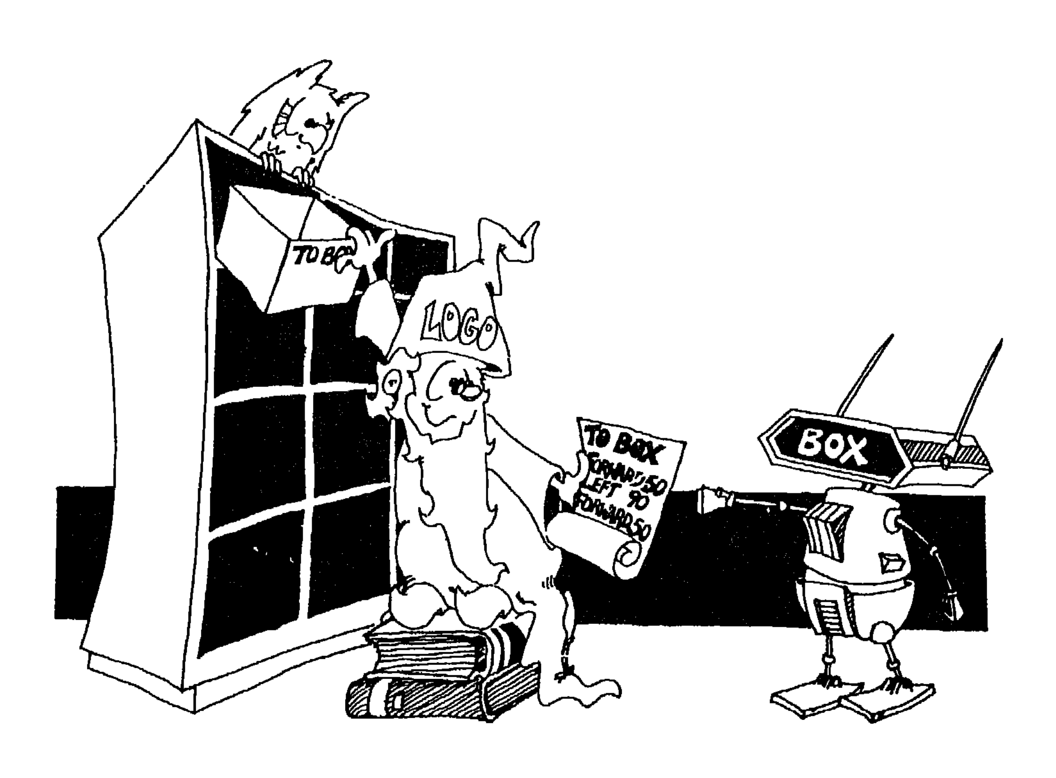
(ب) الروبوط EDIT يختزن كل ما نكتبه من أوامر وكل ما نضيفه من تعديلات بصفحة المحرر .



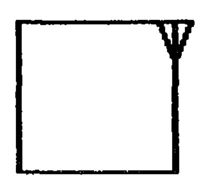
(جـ) عندما نخرج من المحرر إلى طور الاوامر فإن الروبوط EDIT يسلم البرنامج BOX إلى الساحر لوجو ليحتفظ به .



(د) يقوم الساحر لوجو بتخزين تعليمات البرنامج BOX فى مكان معروف يسهل استرجاعها منه . لذلك نجده يضعها فى علبة مكتوب عليها "TO BOX" .



(هـ) عندما نعطى الأمر BOX يقوم الساحر لوجو باسترجاع التعليمات المخزنة فى العلبة : "TO BOX"



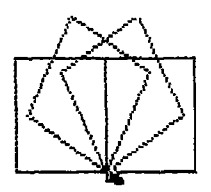
نتيجة تنفيذ الأمر BOX

وهذا تلخيص بالأوامر التى تعلمناها حتى الآن مع الاختصارات الممكنة :

الأمــر	الاختصـار	الوظيفة
TO	–	لبدء تعليم الكومبيوتر .
END	–	إنتهاء تعليم للكومبيوتر .
EDIT	ED	استدعاء محرر لوجو .
ERASE	ER	لمسح برنامج أو أكثر .
ERALL	–	لمسح كل البرامج .

● اكتب لغة الكومبيوتر بنفسك :

بعد نجاحنا فى تلقين الكومبيوتر الأمر الجديد BOX يمكننا أن نكتب لغة جديدة تحتوى على أوامر ليست موجودة فى لغة لوجو أصلاً ولكنها موجودة فى ذاكرة الكومبيوتر الذى نعمل عليه .



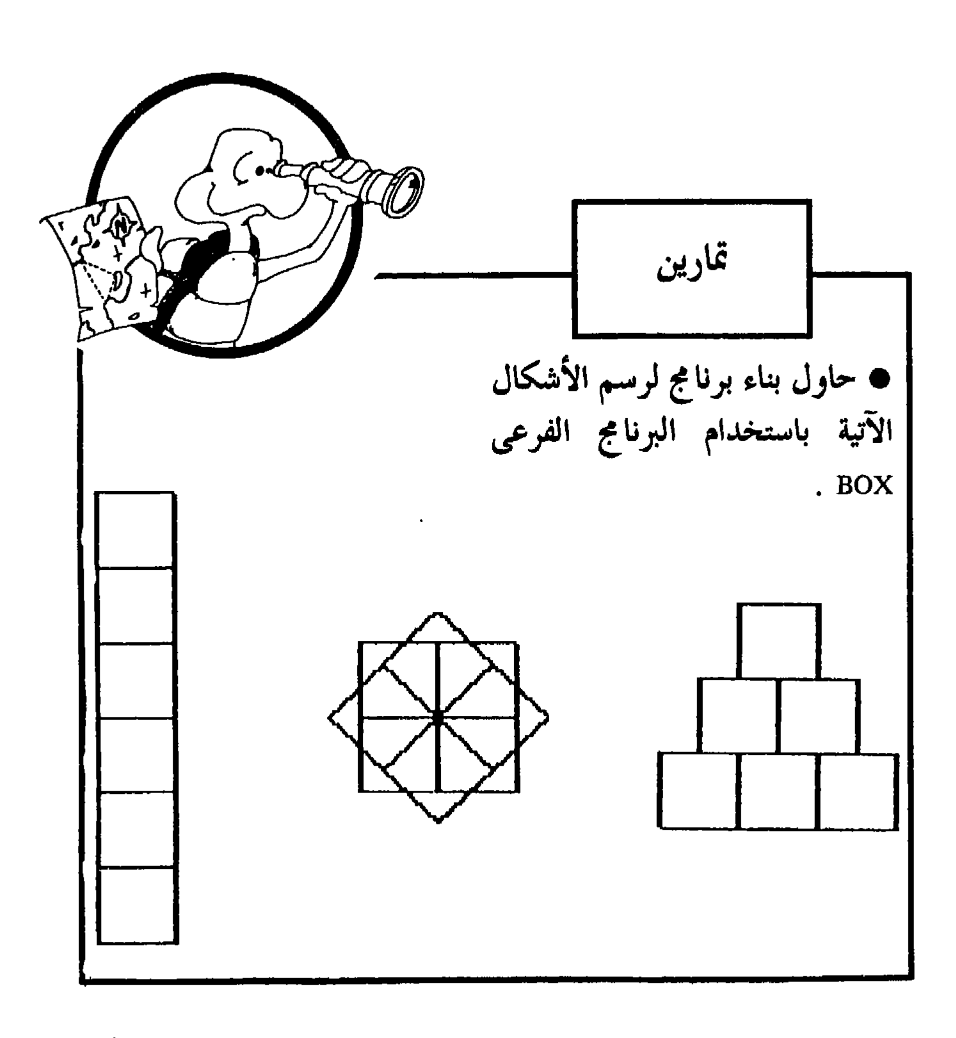
شكل مرسوم باستخدام الأمر BOX

فالرسم الموجود بالشكل تم تنفيذه بالبرنامج الآتى :

PEPEAT 4 [BOX LT 30]

وهنا قد ظهرت كلمة BOX ضمن برنامج آخر متداخلة مع أوامره وكأنها جزء من لغة لوجو . فما معنى ذلك ؟

فى الحقيقة أنه طالما كان البرنامج BOX مخزناً فى ذاكرة الكومبيوتر فيمكنك استخدامه كما شئت سواء كأمر مباشر أو كأمر داخل برنامج آخر كهذه الحالة . وعندما يستخدم البرنامج BCX بهذه الصورة فإنه يسمى برنامجاً فرعياً (subprocedure) ويسمى البرنامج الذى يحتويه البرنامج الذى يحتويه بالبرنامج الرئيسى (superprocedure) .



● عرض البرامج على الشاشة PO, POALL, POTS

● إذا أعطيت الأمر :

PO "BOX

فإن الكومبيوتر يعرض أمامك كل خطوات البرنامج BOX على الشاشة بحيث يمكنك مراجعتها مراجعة سريعة بدون الدخول فى طور التحرير . والأمر PO هو اختصار للعبارة (PRINT OUT) .

● وإذا كان فى الذاكرة أكثر من برنامج فيمكنك عرضهم جميعاً بأمر واحد هو :

POALL

● كما يمكنك عرض عناوين البرامج فقط (أسمائها) بالأمر :

POTS

والكلمة POTS هى اختصار العبارة (PRINT OUT TITLES) .

● إمكانات أخرى للمحرر ED, ED [....]

● علمنا أنه يمكن استدعاء برنامج ما على شاشة المحرر بكتابة الأمر EDIT أو الصورة المختصرة ED يليها اسم البرنامج مثل :

ED "BOX

● يمكن أيضاً إعطاء الأمر EDIT بدون أى اسم معه وفى هذه الحالة يستدعى المحرر آخر برنامج سبق استدعاؤه

● من الممكن أيضاً استدعاء أكثر من برنامج وإصلاحهم معاً بوضع هذه الأسماء فى قائمة كالآتى :

ED [BOX STAR PYRAMD]

والأسماء التى بداخل القوسين المربعين هى أسماء البرامج المطلوب إحضارها على شاشة المحرر لإصلاحها .

● حفظ البرامج فى الذاكرة الخارجية واستدعاؤها :
SAVE, LOAD, DIR, ERASEFILE

كما نعلم أن ذاكرة الكومبيوتر التى نتحدث عنها دائماً هى ذاكرة مؤقتة أو متطايرة بمعنى أن كل ما هو موجود بها يتطاير منها بمجرد إطفاء الجهاز . فإذا أطفأنا الكومبيوتر ذهبت جميع البرامج وذهبت لغة لوجو أيضاً . والاسم الدقيق للذاكرة هو الذاكرة الداخلية .

ولحفظ البرامج بصفة مستديمة يلزم حفظها فى الذاكرة الخارجية التى تتميز بالدوام والثبات وهى القرص المغنطيسى أو الشريط الكاسيت .

ولحفظ برنامج ما يدعى SALLYنعطى الأمر :

SAVE "SALLY

ولاستدعاء هذا البرنامج من القرص أو الكاسيت نعطى الأمر :

LOAD "SALLY

وهذه الأوامر تنطبق على القرص المغنطيسى للكومبيوتر IBM وكذلك على الشريط الكاسيت للأجهزة الصغيرة . أما أوامر القرص المغنطيسى للأجهزة الصغيرة فقد تختلف ويرجع إليها بدفتر اللغة .

يمكن أيضاً عرض جميع البرامج المخزنة على القرص باستخدام الأمر :

DIR

وفى بعض الأجهزة الأخرى يتم عرض البرامج باستخدام الأمر :

CATALOG

كما يمكن مسح البرنامج من القرص المغنطيسى باستخدام الأمر .

ERASEFILE "SALLY

كما يمكن حفظ واستدعاء الصور المرسومة بلغة لوجو مباشرة بالأوامر :

SAVEPIC "BOXES للحفظ

LOADPIC "BOXES للاستدعاء

اسم الصورة

وبصفة عامة فإن أوامر الحفظ ليست قياسية ويلزم الرجوع لكتاب الجهاز المعيّن .

● **تشغيل جهاز الطباعة** **SAVE "LPT1, DRIBBLE, NODRIBBLE PRINTON, PRINTOFF, OUTDEV**

هذه هى أوامر الطباعة على جهاز الطباعة PTINYRT **للكومبيوتر IBM** :

SAVE "LPT1

هذا الأمر يطبع كل شىء فى حيز العمل على الطابعة.أما الأمر التالى فهو يجعل الطابعة على اتصال دائم بالكومبيوتر :

DRIBBLE "LPT1

لذلك يقال أن الطابعة فى حالة « تصنّت » ! فإذا كتبت الأمر الآتى :

PRINT [hello you]

فإن العبارة hello you تطبع فوراً على الطابعة .

وإذا أعطيت الأمر :

POTS

فإن أسماء البرامج التى فى الذاكرة الداخلية تُطبع فوراً .

وإذا كتبت الأمر :

PO "BOX

فإن سطور البرنامج BOX تُطبع على الورق .

وهكذا ..

والآن نعطى الأمر المضاد الذى يوقف عمل الطابعة وهو :

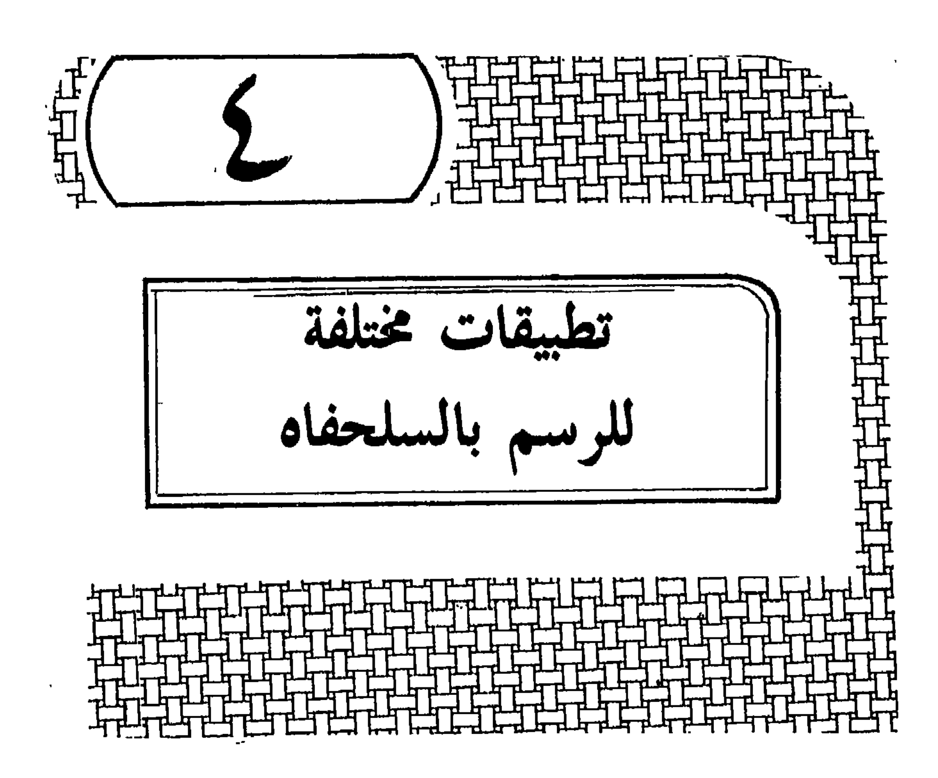
NODRIBBLE

ملاحظة

أما أوامر الطبع للأجهزة الصغيرة فقد تختلف كثيراً وعلى سبيل المثال فالكومبيوتر سنكلير يستخدم الأمر PRINTON كنظير للأمر DRIBBLE ويستخدم PRINTOFF كنظير للأمر NODRIBBLE أما لنسخ الشاشة كلها فهو يستخدم الأمر COPYSCREEN وهذا يناظره فى الكومبيوتر IBM الضغط على الأزرار :

Shift ↑ + Prtscr.

أما مع الكومبيوتر apple II فالأمر OUTDEV 1 يستخدم لتشغيل الطابعة والأمر OUTDEV 0 يستخدم لفصل الطابعة . وبصفة عامة يرجع لكتاب اللغة .

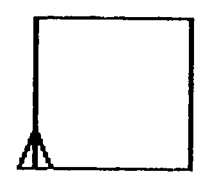

٤
تطبيقات مختلفة
للرسم بالسلحفاه

● استخدام البرامج الفرعية :

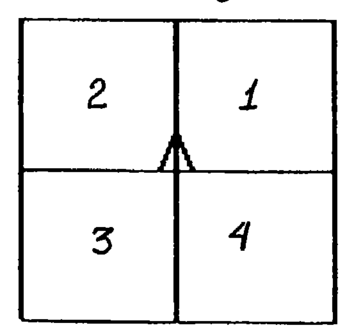
إذا كنت قد احتفظت بنسخه من برنامج رسم المربع يمكنك الآن أن تستدعيها من الشريط الكاسيت أو من القرص المغنطيسى فإذا لم تكن قد فعلت فاكتب هذا البرنامج ؛ وتأكد من أنه يرسم الشكل المطلوب الموضح بعد :

```
TO SQUARE
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
END
```

البرنامج الفرعى



نريد الآن استخدام هذا البرنامج كبرنامج فرعى لرسم أربعة مربعات متلاصقة ببعضها البعض كما فى الشكل التالى :



هذا يمكن أن يتم بإنشاء برنامج جديد (رئيسى) يستخدم البرنامج الفرعى SQUARE بالطريقة المناسبة . وهى رسم المربع رقم 1 ثم إدارة السلحفاه ٩٠:

درجة جهة اليسار (أو جهة اليمين) ورسم المربع رقم 2 ثم إدارتها ٩٠ درجة أخرى لليسار ورسم المربع رقم 3 وهكذا بذلك يحتوى البرنامج الرئيسى على الأوامر التالية :

```
TO BOXES
SQUARE
LEFT 90
SQUARE
LEFT 90
SQUARE
LEFT 90
SQUARE
LEFT 90
END
```

البرنامج الرئيسى

وبالطبع سوف يقول أصدقاؤنا الذين استوعبوا الأجزاء السابقة :

« ولكن هذه طريقة بدائية لكتابة البرامج .. أين REPEAT وأين الاختصارات لتغنى عن هذا كله ؟ » وهذا حق .

فمن الممكن كتابة البرنامج الأول كالآتى :

REPEAT 4 [FD 50 RT 90]

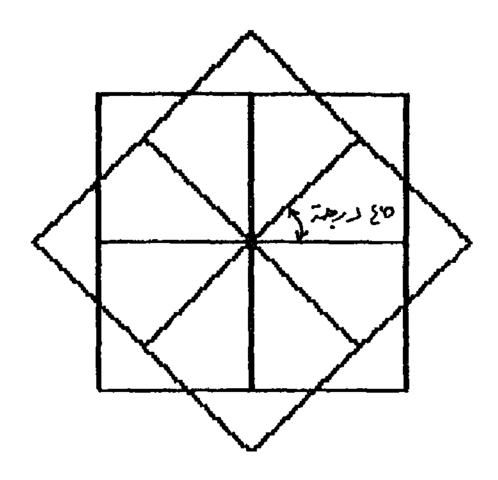
والبرنامج الرئيسى يصبح كالآتى :

REPEAT 4 [SQUARE LT 90]

ولكن لا بأس من استخدام البرامج البدائية أحياناً حتى يتابعنا جميع القراء وسوف نقدم الصورة المختصرة من آن إلى آخر .

والآن بإعطاء الأمر BOXES فإن البرنامج الرئيسى BOXES يستدعى البرنامج الفرعى SQUARE ليرسم المربّعات الأربعة . ثم تعود السلحفاه إلى بيتها وفى اتجاهها الأصلى .

نريد الآن أن نستخدم البرنامج الرئيسى BOXES كبرنامج فرعى لرسم الشكل الموضح بعد والمكون من تكرار للشكل السابق بزوايا مختلفة :

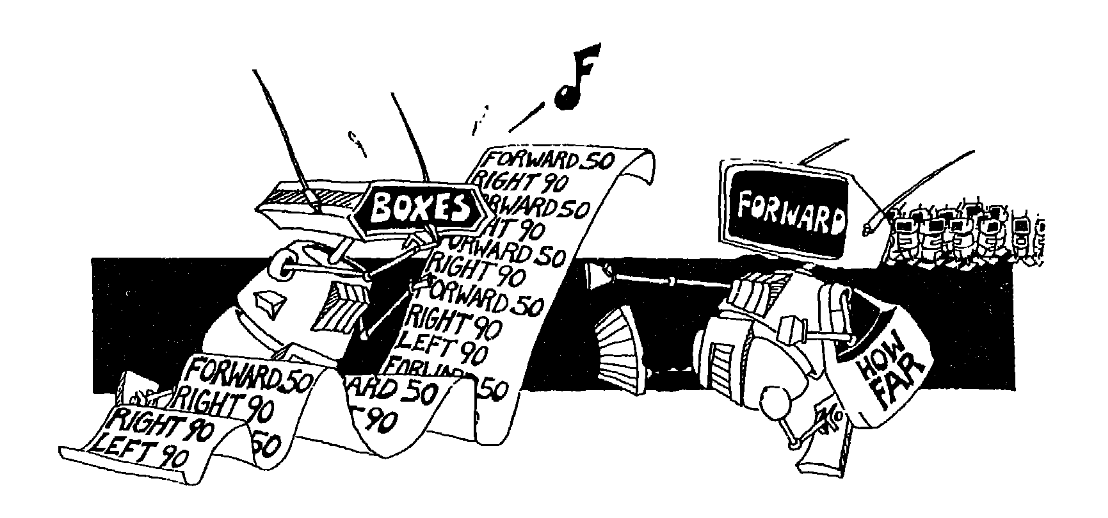


هذا الشكل عبارة عن الشكل الأصلى BOXES مرسوماً مرتين . ولكن فى المرة الثانية رسمناه مائلاً بزاوية ٤٥ درجة .

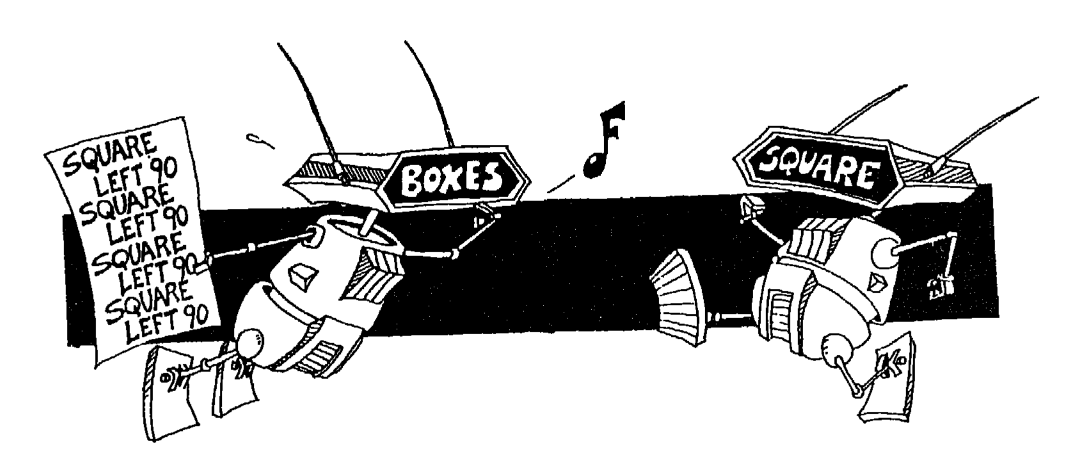
لنكتب إذن هذا البرنامج الرئيسى وليكن إسمه STAR ليستدعى البرنامج الفرعى BOXES ويستخدمه فى رسم الشكل .

```
TO STAR
BOXES
RIGHT 45
BOXES
END
```

تصور الآن لو لم نستخدم البرامج الفرعية ! كم كان البرنامج الأخير سيكون طويلاً ومرهقاً فى كتابته بل مرهقاً فى قراءته وفهمه أيضاً . وهذا هو الروبوط BOXES حائر ما بين الأوامر الكثيرة التى يحتوى عليها برنامج بلا برامج فرعية .



كم يكون العمل مرهقاً بدون البرامج الفرعيّة !



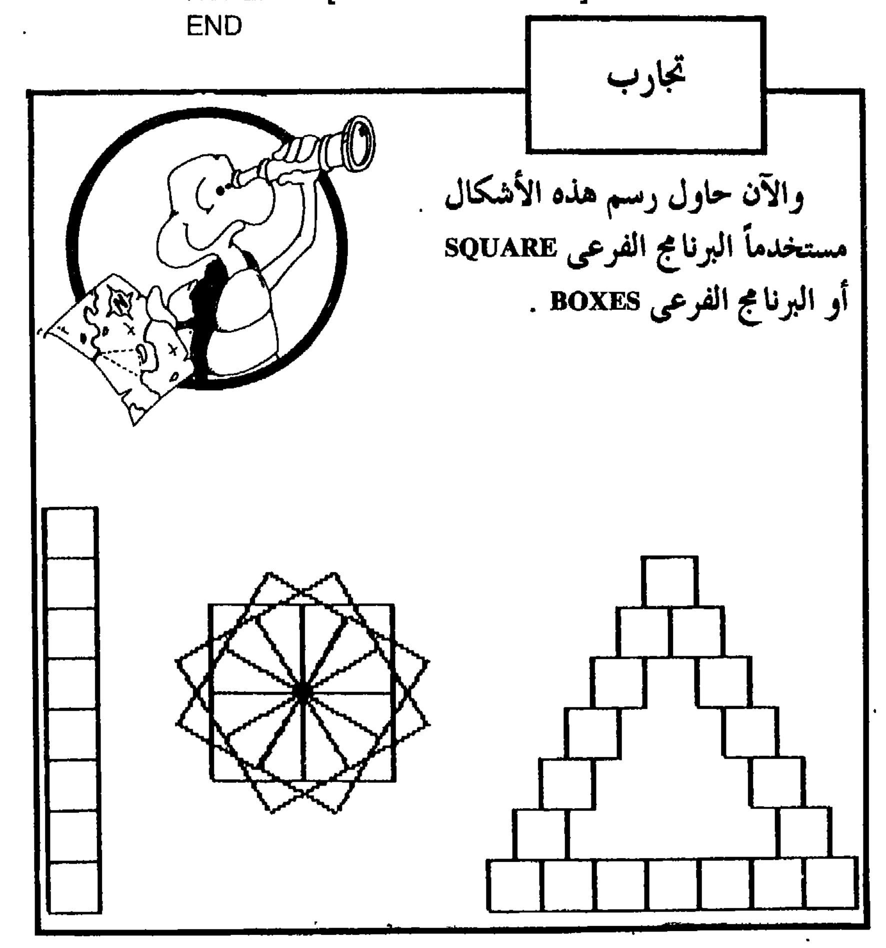
وكم يصبح العمل سهلاً باستخدام البرامج الفرعية

وهذه هى الصورة المختصرة للبرامج الثلاثة (مع إضافة الرقم 1 للأسماء للتمييز) :

```
TO SQUARE 1
REPEAT 4 [FORWARD 50 RIGHT 90]
END

TO BOXES1
REPEAT 4 [SQUARE1 LEFT 90]
END

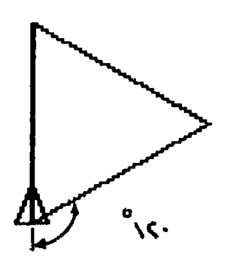
TO STAR1
REPEAT 2 [BOXES1 RIGHT 45]
END
```

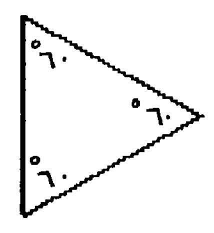


● رسم المثلثات :

عندما رسمنا شكلاً مربعاً استخدمنا الدوران بزاوية ٩٠ درجة وبذلك فإن الزاوية الدائرية الكاملة (٣٦٠ درجة) اتسعت لعدد ٤ مربعات لأن ٤ × ٩٠ = ٣٦٠ .

عندما نرسم مثلثاً متساوى الأضلاع فإن كل زاوية من زواياه تكون ٦٠ درجة . أى أن السلحفاه يجب أن تستدير ١٢٠ درجة بعد انتهائها من رسم كل خط .





إذن يمكن كتابة برنامج المثلث كالآتى :

```
TO TRIANGLE
REPEAT 3 [FORWARD 80 RIGHT 120]
END
```

التكرار

زاوية الدوران

ومن هذا الشكل يمكن أن ننشيء العديد من الأشكال باستخدامه كبرنامج فرعى . وسوف نتعرض لكثير من تطبيقات المثلثات فى الأجزاء القادمة . ولكننا الآن نريد أن ننتقل إلى نوعية أخرى من المثلثات وهى النجوم .

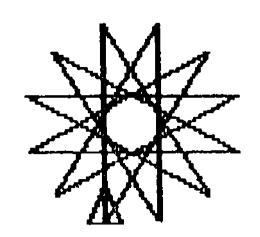
● رســم النجــوم :

لنبدأ محاولة رسم نجمة بالسطور الآتية وباستخدام الدوران ١٥٠ درجة .

```
FORWARD 50
RIGHT 150
FORWARD 50
RIGHT 150
```







خطوات رسم النجمة

وبتكرار هذا الرسم مرة بعد مرة نحصل على الشكل المطلوب .

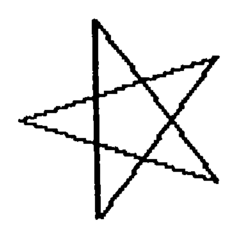
والواقع أنه يمكن تجربة عدة زوايا مختلفة بخلاف ١٥٠ درجة وملاحظة ما يحدث على الشاشة ثم تقليل أو تكبير الزاوية بحيث تلتقى نقطة البداية مع نقطة النهاية . أما عدد مرات التكرار فهى تعبر عن عدد الخطوط المكونة منها النجمة وهى ١٢ خطأ (فى حالتنا هذه) أى أن البرنامج يمكن أن يكون كالآتى :

```
TO STAR 150
REPEAT 12 [FORWARD 100 RIGHT 150]
END
```

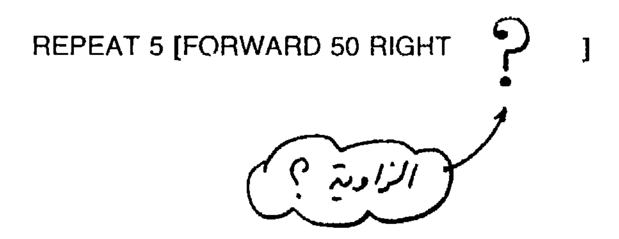
(عدد الخطوط) (زاوية الدوران)

هذه نجمة ذات ١٢ ضلعاً .

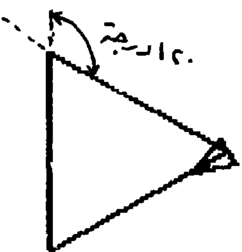
فماذا لو أردنا رسم النجمة العربية ذات الخمسة أضلاع ؟



من المتوقع أن يكون البرنامج بالصورة التالية :



ومبدئياً فإن هذه الزاوية لابد أن تقع بين ٩٠ درجة و ١٨٠ درجة ولكن لإيجادها بدقة فهذا يحتاج إلى تجارب كثيرة . فلنبدأ التجربة بالزاوية ١٢٠ درجة .

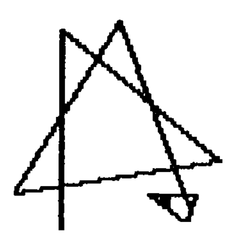



```
REPEAT 5 [FORWARD 80 LEFT 120]
```

إن نقطتى البداية والنهاية تلتقيان ولكن الشكل الناتج مثلث كما نرى ! إن الزاوية ١٢٠ درجة تبدو صغيرة جداً لأنها لا تجعل الخطوط تتقاطع .

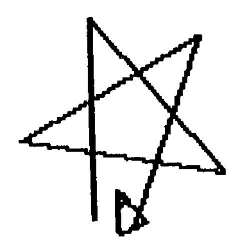
فلنجرب إذن الزاوية ١٣٠ درجة .

```
REPEAT 5 [FORWARD 80 LEFT 130]
```



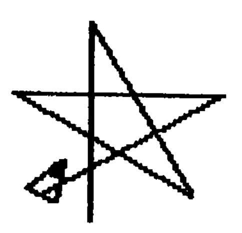
إن التقاطع يحدث ولكن الشكل مفتوح لا تلتقى بدايته مع نهايته . فلنجعلها ١٤٠ درجة .

```
REPEAT 5 [FORWARD 80 LEFT 140]
```



مازال الشكل مفتوحاً لكنه يبشر بقرب النجاح . لنجرب الآن ١٥٠ درجة .

```
REPEAT 5 [FORWARD 80 LEFT 150]
```



لقد حصلنا على الشكل المقفول .. ومع ذلك فنقطة التقاء البداية مع النهاية حدثت « متأخرة » قليلاً . إذن هذه الزاوية أكبر من المطلوب . فالزاوية

المطلوبة تقع بين ١٤٠ درجة و ١٥٠ درجة . لكى تكمل التجارب لابد من إخفاء السلحفاه حتى تشاهد نقطة التلاقى بدقة . ربما يحتاج الأمر إلى تجارب كثيرة .. والبديل الوحيد أن نلجأ إلى التخمين الرياضى .. فى الفقرة القادمة ..

* هل هناك بديل للتجربة والخطأ ؟

القيمة التى نبحث عنها هى 144 .

ولكنها قيمة صعبة المنال مع المحاولة والخطأ .

وفى الواقع أن هناك طريقة تفكير رياضية تغنى عن هذه المحاولات العشوائية .

إن السلحفاه فى رحلتها لرسم أى شكل مقفول ، تدور خلال زاوية مقدارها أحد مضاعفات العدد 360 (بمعنى أن تكون 360 أو 360 × 2 أو 360 × 3 وهكذا ..) وذلك قبل أن تصل إلى نقطة البداية .

وفى حالة النجمة ذات الخمسة أضلاع فإن السلحفاه تدور خمس مرات . فليكن إذن مقدار كل زاوية دوران هو (360/5) وهذه هى التجربة الأولى . فإذا لم تنجح لنجرب الرقم الآخر وهو (360/5) × 2 وهكذا ...

والرقم الذى سينجح هو الأخير أى 144 درجة .

وعلى ذلك يكون البرنامج هو :

REPEAT 5 [FD 80 LT 144]

ولنسم هذا البرنامج الفرعى STAR144 دلالة على الزاوية المحيرة 144 .



تجربة

استنتج الآن الزاوية التى تدور بها السلحفاه لرسم النجمة ذات الأضلاع التسعة الموضحة بالرسم واكتب البرنامج الذى يرسمها .

* المــرآة **mirror**

لو أنك استبدلت كل دوران لليمين بدوران لليسار (والعكس) فإنك تحصل على نفس الشكل المرسوم ولكنه يبدو كما لو كان معكوساً على سطح مرآة .

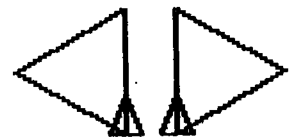
فالمربع square سبق لنا أن رسمناه بالبرنامج الآتى :

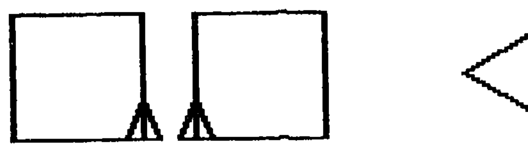
REPEAT 4 [FD 50 RT 90]

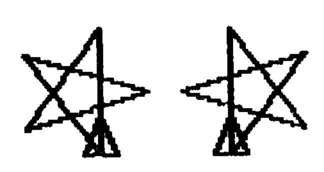
والمربع المنعكس على المرآة يكون برنامجه كالتالى :

REPEAT 4 [FD 50 LT 90]

ويطلق على المربع الأخير صورة المربع على المرآة (mirror image) والشكل التالى يوضح بعض الرسومات التى رسمناها وصورة كل رسم على المرآة . وعليك باستنتاج البرامج المناظرة .







وهناك طريقة أخرى للحصول على الصورة وهى استبدال الزاوية المقررة للدوران ولتكن n بالزاوية (360-n) مع بقاء جهة الدوران كما هى . فمثلاً للحصول على صورة المربع نستبدل الزاوية 90 بالزاوية 270 وللحصول على صورة المثلث نستبدل الزاوية 120 بالزاوية 240 وهكذا ..

● فكرة لهواة البحث والتنقيب .. فقط !

البرنامج الذى عرضناه لرسم النجمة ذات الإثنى عشر ضلعاً استخدمنا فيه الزاوية ١٥٠ درجة . فإذا أردنا أن نعرف كيفية حساب هذه الزاوية . فهى حسب القاعدة تكون أحد مضاعفات الزاوية (360/12) أى الزاوية (30) درجة . وبالتجربة نجد أنها 5 * (360/12) أى أنها المحاولة رقم 5 .



والسؤال هنا :

ماذا عن الأشكال الأخرى التى تناظر المحاولة الأولى والثانية والثالثة والرابعة التى تناظر الزاويا 4*(360/12) ، 3*(360/12) .. وهكذا ..

ولنعتبر مثلاً الشكل الذى نحصل
عليه عند الزاوية 3*(360/12) أى ٩٠
درجة .

إن هذا الشكل عبارة عن مربع وهذا بديهى لأن الدوران بزاوية ٩٠ درجة مرّة بعد مرّة يرسم مربعاً .

ولكن البرنامج يرسم هذا المربع ٣ مرات لأن أمر التكرار يعقبه الرقم ١٢ .

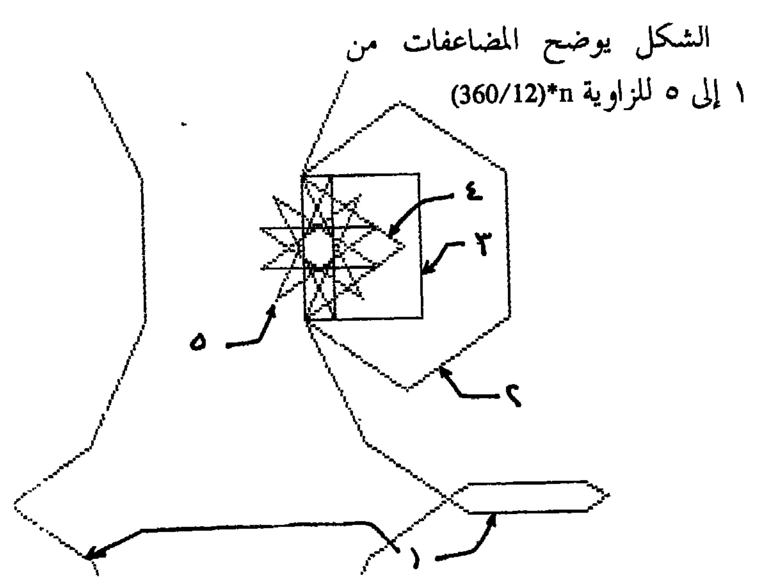
REPEAT 12 [FD 50 RT 90]

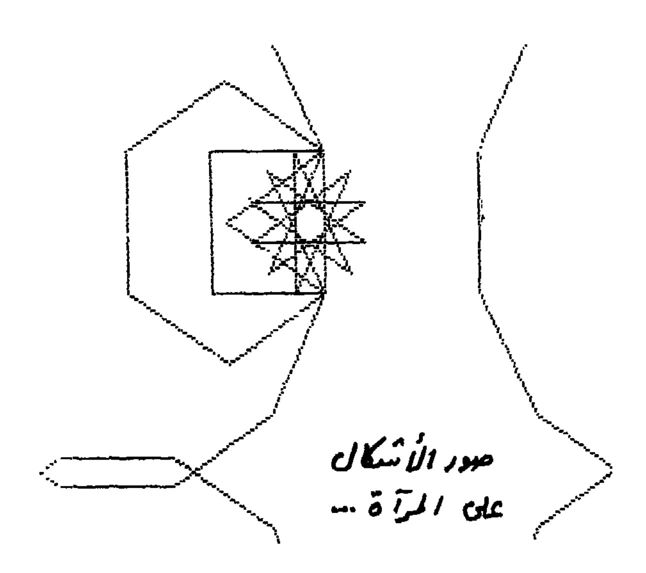
ولو أكملنا التجارب حتى المضاعفات رقم ٦ ، ٧ ، ٨ ، ٩ ، ١٠ ، ١١ .. لوجدنا أن الأشكال التى نحصل عليها تتكرر وأحياناً تنقلب إلى صور على المرآه .

والآتى بعد هو بعض الأشكال التى نحصل عليها بتغيير الرقم المضروب فى (360/12) وكذلك صورها على المرآة .

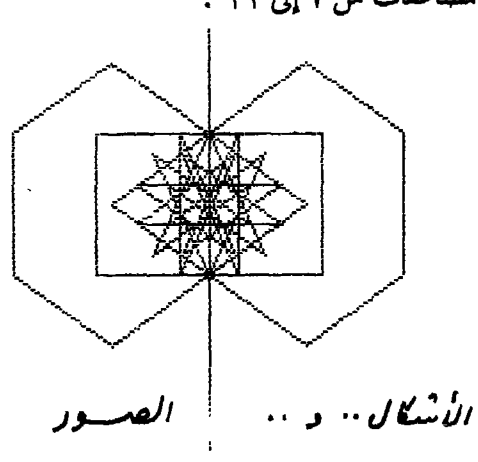
جرب البرنامج وحاول أن تعلل لهذه الأشكال والصور التى تحصل عليها فى ضوء الدراسة السابقة .

الشكل يوضح المضاعفات من
١ إلى ٥ للزاوية n*(360/12)





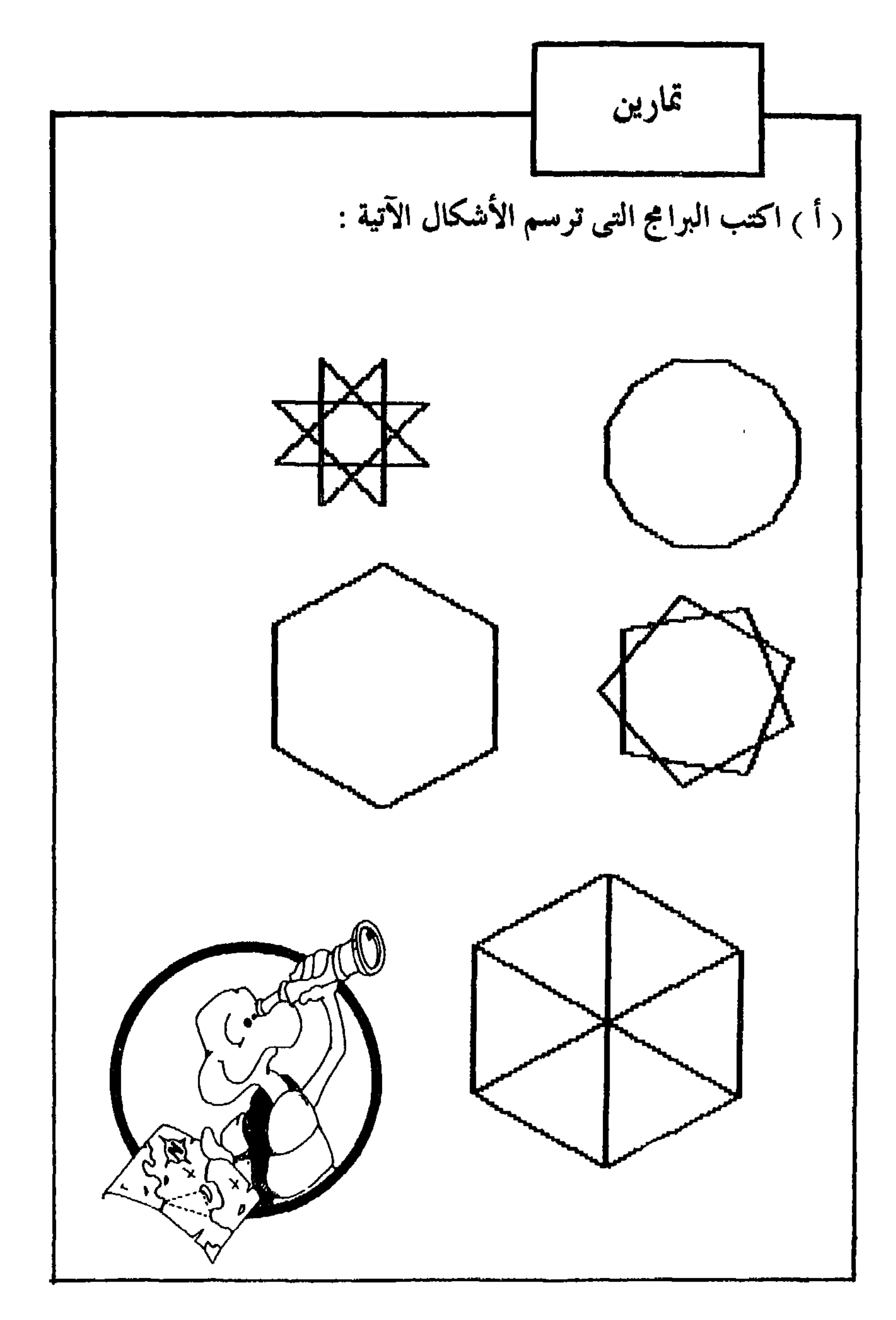
الشكل يوضح الصور المناظرة للأشكال السابقة على المرآة وهى تناظر المضاعفات من ٧ إلى ١١ .



الشكل يوضح المضاعفات من ٢ إلى ٥ وصورها على المرآة والخط الفاصل بين المجموعتين يوضح المرآة وهو نفسه أحد الأشكال الناتجة عن القيمة (n = 0) .

تمارين

(أ) اكتب البرامج التى ترسم الأشكال الآتية :

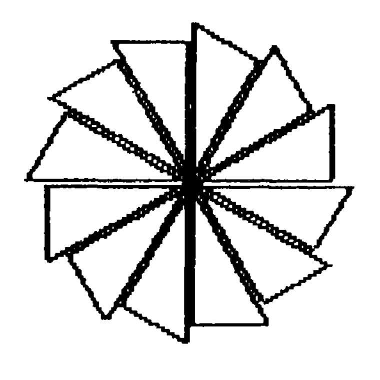


(ب) إذا أعطيت البرنامج التالى الذى يرسم الشكل الموضح بجواره :



```
FORWARD 50
LEFT 90
FORWARD 20
LEFT 120
FORWARD 30
```

فحاول رسم الشكل الآتى باستخدام البرنامج :



* الدوائر من المبادىء الأولية :

للكومبيوتر IBM مجموعة برامج خاصة يمكن استدعاؤها من القرص المغنطيسى "LWIL procedures" وهذه البرامج تمكّن من إعطاء أوامر جديدة للدوائر والمنحنيات مثل :

RCIRCLE n (لرسم دائرة على اليمين نصف قطرها n)

LCIRCLE n (لرسم دائرة على اليسار نصف قطرها n)

RARC n (لرسم قوس على اليمين من دائرة نصف قطرها n)

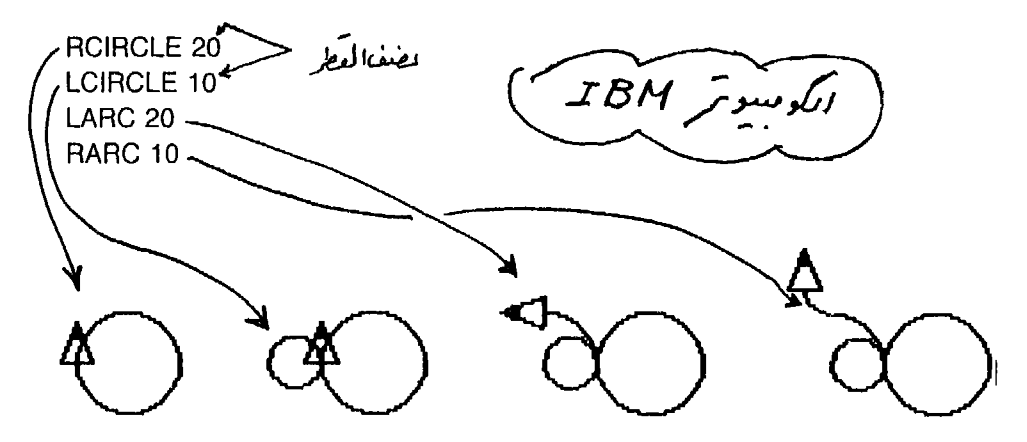
LARC n (لرسم قوس على اليسار من دائرة نصف قطرها n)

ويتم استدعاء هذه الأوامر بإعطاء الأمر :

LOAD "CIRCLES

مع وضع القرص المغنطيسى LWIL فى المكان المخصص له عند إعطاء الأمر والملحق (١) والملحق (٢) فى نهاية الكتاب بهما شرح وافٍ لهذه البرامج كما يمكن نسخها وحفظها على القرص المغنطيسى لاستخدامها عند الحاجة إليها بنفس الأسماء مع أى كومبيوتر آخر .

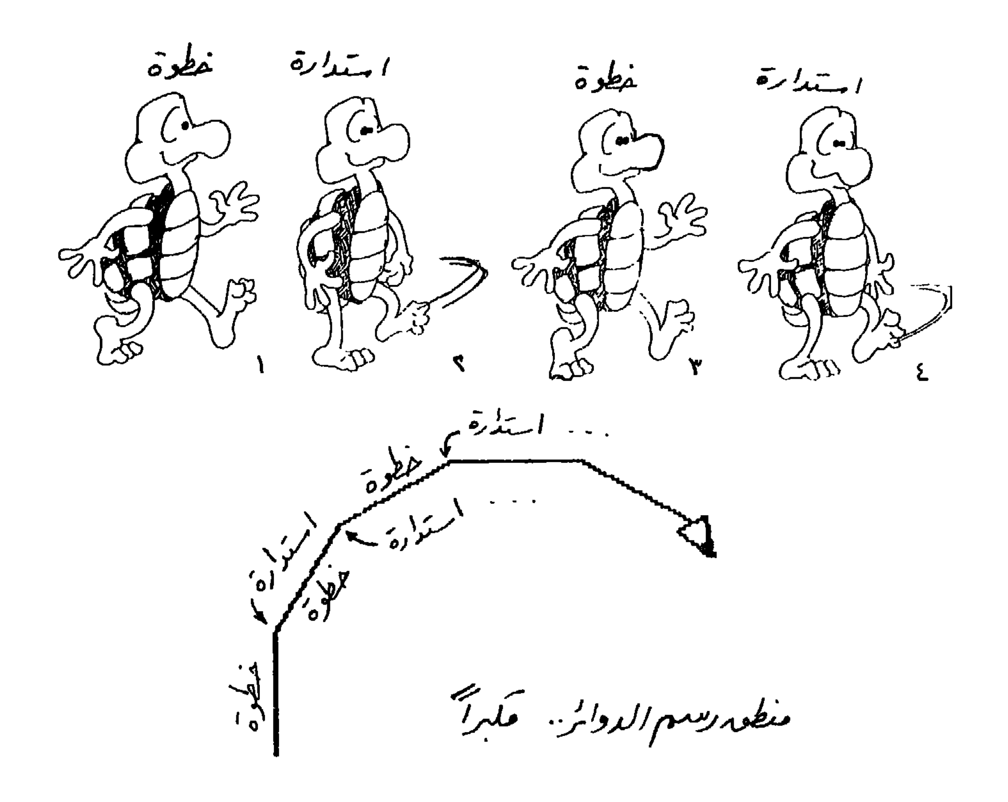
والمثال الآتى يوضح استخدام هذه الأوامر :



ومع ذلك فليس هذا هو موضوعنا :

فنحن نرغب فى بناء البرنامج الذى يرسم دائرة باستخدام المبادىء التى تعلّمناها . فما الفكرة فى رسم الدائرة ؟ إن السلحفاه تتقدم خطوة وتستدير مرة وتتقدم خطوة أخرى وتستدير مرة ثانية وهكذا .. خطوة صغيرة يعقبها استدارة صغيرة .

وهذا ما تفعله السلحفاه فى الشكل التالى وهى ترسم الدائرة .



بلغة لوجو يكون البرنامج كالآتى :

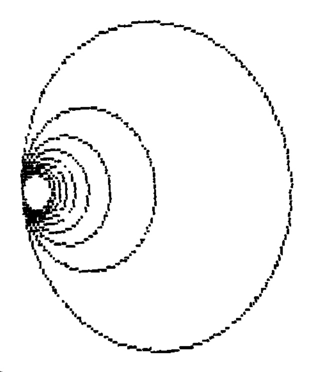
REPEAT 360 [FD 1 RT 1]

هذا يرسم دائرة على اليمين فإذا أردنا أن تكون الدائرة على اليسار نعدّل (RT) لتكون (LT) فيصبح البرنامج :

REPEAT 360 [FD 1 LT 1]

وبالتحكم فى مقدار الدوران (RT) أو (LT) يمكن الحصول على أقطار مختلفة .

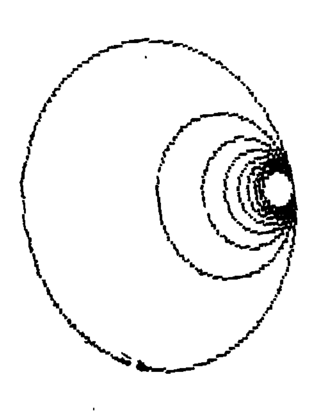
فالشكل الآتى بعد مرسوم بالبرنامج الأول (الدائرة على اليمين) . والدائرة الكبيرة تناظر (RT 1) أما الدوائر الأصغر فهى تناظر (RT 2) ، (RT 3) ،.. وهكذا حتى (RT 9) .



(منفذ على كومبيوتر سنكلير)

مجموعة دوائر على اليمين (RT)

وباستخدام الدوران لليسار LT بدلاً من RT نحصل على مرآة الصورة كما في الشكل التالى :



(منفذ على كومبيوتر سنكلير)

مجموعة دوائر على اليسار (LT)

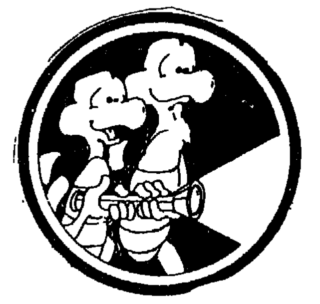
كما يمكن جعل الدوائر تظهر أعلى أو أسفل صفحة الشاشة بإدارة السلحفاه مبدئياً قبل الرسم بمقدار ۹۰ درجة إلى أحد الجهتين اليسرى أو اليمنى .

والرسم التالى بعد يوضح مجموعة من الدوائر تم رسمها بعد إعطاء الأمر LT 90 وقد أظهرنا فيها السلحفاه حتى يظهر اتجاهها الأصلى الذى بدأت منه

وانتهت إليه . والرسم منفذ على الكومبيوتر سنكلير ويمكن تنفيذه على أى كومبيوتر آخر ، لا فرق .



فلاش

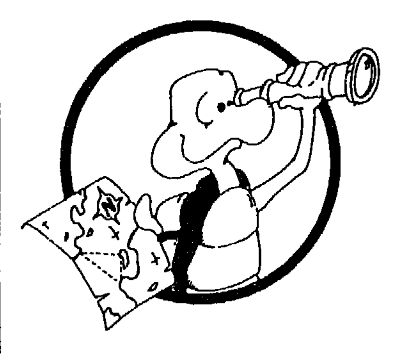


إذا استخدمت الدوران 2 RT فى برنامج الدائرة فسوف تلاحظ أن السلحفاه ترسم الدائرة مرتين متطابقتين . وفى هذه الحالة يكفى تكرار البرنامج بمقدار (360/2) أى 180 مرة .

وكذلك بالنسبة للدوران (3 RT) يمكن قسمة عدد مرات التكرار على 3 لتصبح 120 درجة وهكذا ..

تمرين

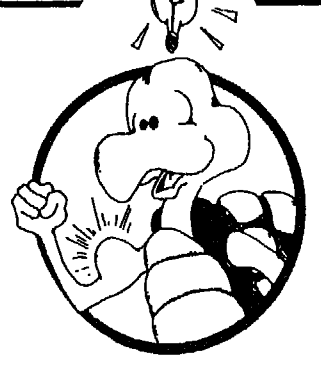




● ارسم مجموعة الدوائر التى تحتل النصف العلوى من الشاشة باستخدام درجات دوران مختلفة .

● اكتب البرنامج (الرئيسى) الذى يرسم مجموعتين من الدوائر على يمين ويسار الشاشة باستخدام البرامج الفرعية المناسبة .

فكـرة



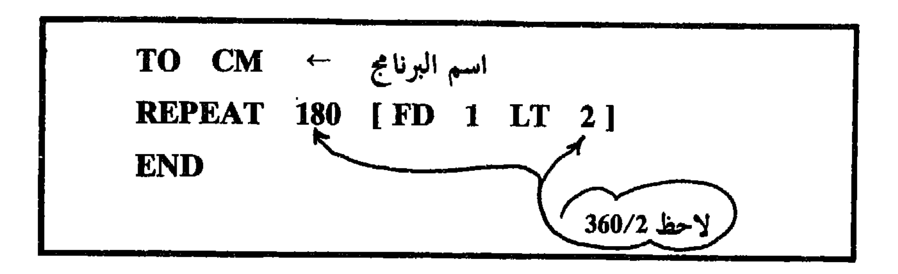
عرفنا تأثير درجة الدوران على نصف قطر الدائرة فكلما كبرت زاوية الدوران كان نصف القطر أقصر أو بمعنى آخر فإن السلحفاه تلف لفة سريعة وتعود إلى بيتها .

هل تستطيع أن تدرس بنفسك تأثير الخطوة FD على مساحة الدائرة ؟ إنها تجارب شائقة (ركز اهتمامك فى نصف القطر ودرجة الدقة معاً) .

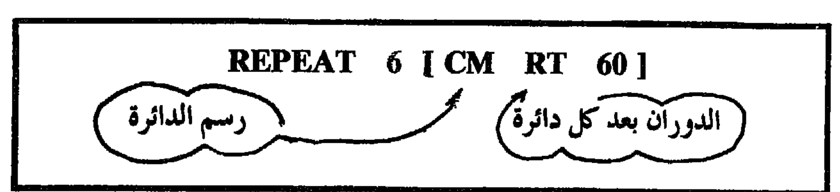
● ارسم ما تشاء من الأشكال باستخدام الدوائر :

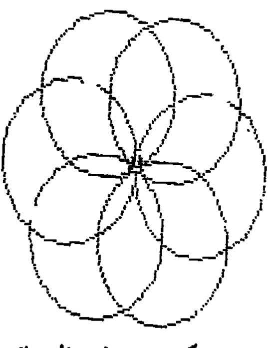
وبتكرار برنامج الدائرة التكرار المناسب يمكن الحصول على ما نشاء من الأشكال الزخرفية التي تعتمد على الدائرة كأساس فى تصميمها .

فالبرنامج التالى يرسم دائرة واحدة على يسار الشاشة :



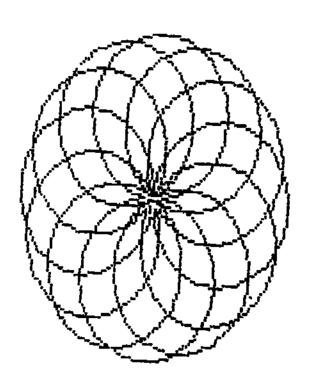
وباستخدام هذا البرنامج كبرنامج فرعى يمكن رسم أشكال زخرفية مختلفة . والأمر الآتى يكرر الدائرة كل ٦٠ درجة (وبالتالى فقد استخدمنا أمر التكرار REPEAT 6) فنحصل على الشكل الموضح بعد :





رسم شكل زخرفى بالدوائر

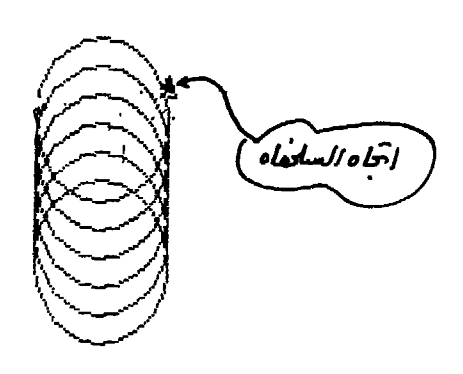
أما البرنامج التالى فهو يكرر رسم الدائرة ١٢ مرة وبين كل دائرة وأخرى تنحرف السلحفاه بمقدار ٣٠ درجة . ونستخدم فيه البرنامج الفرعى السابق CM :



```
TO CIRCLES
REPEAT 12 [ CM RT 30 ]
END
```

أما البرنامج الآتى فهو يرسم مجموعة من الدوائر الرأسية التى تعطى شكل الاسطوانة المفرغة ويتم ذلك بالتحرك إلى أعلى عشر خطوات (FD 10) بعد رسم كل دائرة مع رفع القلم (PU) ثم خفضه مرة أخرى (PD) والبدء فى رسم الدائرة التالية :

```
TO CYLINDER
REPEAT 6 [ CM PU FD 10 PD ]
END
```

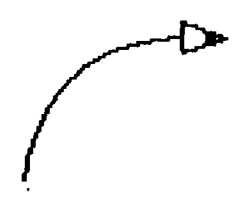


● **المنحنيات والأقواس :**

أمّا القوس فهو دائرة غير كاملة بمعنى أن عدد مرات التكرار أقل من أن يكمل الدائرة .

وهذا الأمر يرسم قوساً يمثل ربع دائرة :

REPEAT 90 [FD 1 RT 1]

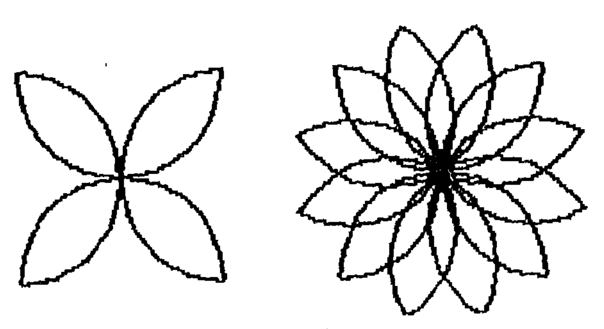


والأمر التالى يرسم قوساً يمثل نصف دائرة :

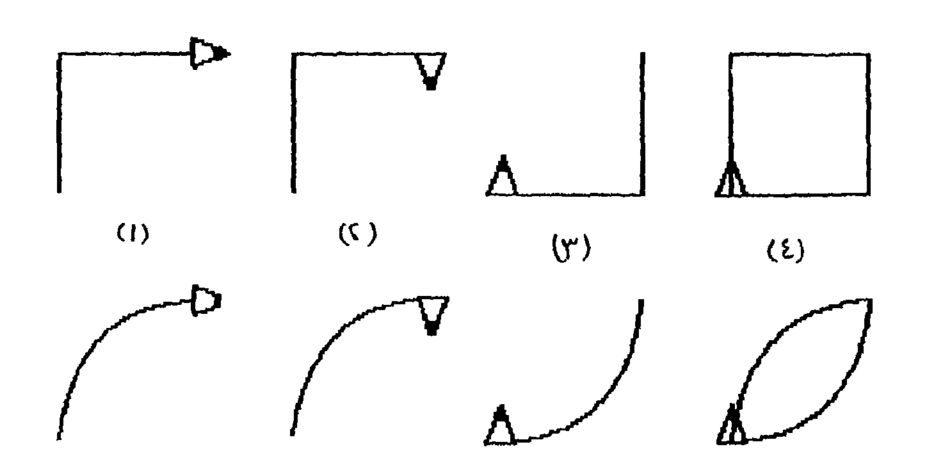
REPEAT 180 [FD 1 RT 1]



وبالتحكم فى زاوية الدوران يمكن الحصول على أشكال مختلفة للمنحنيات سواء من ناحية جهة الدوران أو مقدار الدوران وهذه بعض الأشكال التى يمكن رسمها بالمنحنيات وهى تعتمد على وحدة رسم أصلية تشبه ورقة الشجرة أو بتلّة الزهرة (PETAL) .



ورسم ورقة الشجرة يشبه كثيراً رسم المربع . تتبّع الرسم التالى الذى يوضح خطوات رسم المربع والخطوات المناظرة لها لرسم ورقة الشجرة .

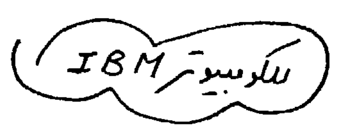


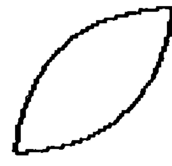
والبرنامج التالى يرسم ورقة الشجرة من المبادىء الأوليّة :

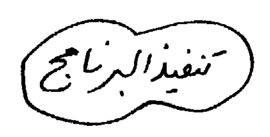
```
TO ARC1
REPEAT 45 [FD 1 RT 2]
RT 90
REPEAT 45 [FD 1 RT 2]
RT 90
END
```

* أما إذا كنت تعمل على **الكومبيوتر IBM** فيمكنك استخدام البرنامج التالى حيث نستخدم الأوامر الخاصة لرسم القوس مثل RARC :

```
TO PETAL
RARC 30
RIGHT 90
RARC 30
RIGHT 90
END
```

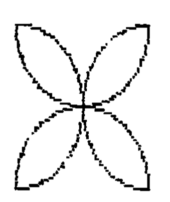


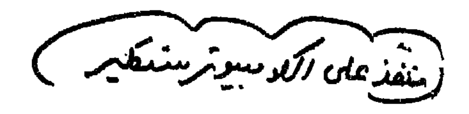




ويمكن استخدام أى من البرنامجين السابقين ARC1 أو PETAL كبرنامج فرعى بداخل البرنامج الرئيسى الآتى لرسم الشكل الموضح مع البرنامج :

```
TO ARC2        اسم البرنامج الفرعى
CS
REPEAT 4 [ARC1 RT 90]
END
```

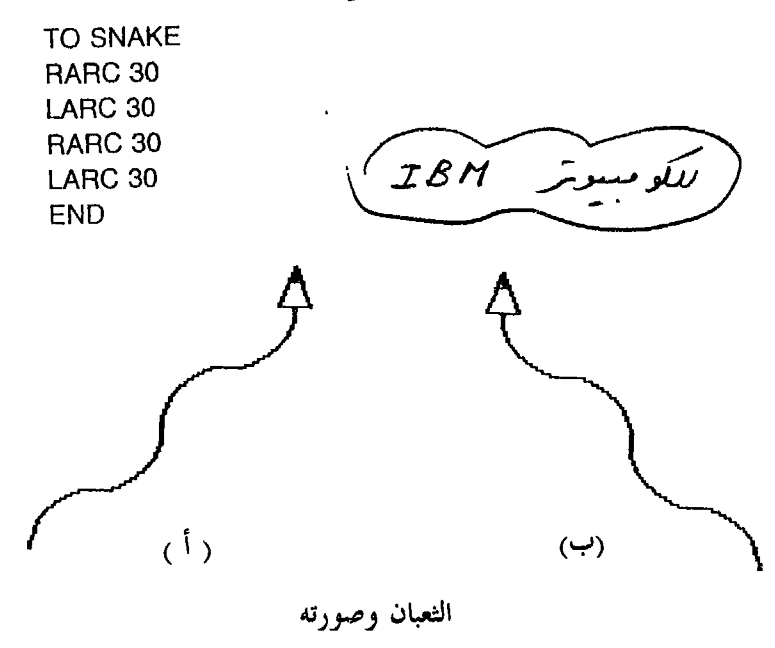




• ارسم ثعباناً بالمنحنيات :

هذا البرنامج يرسم ثعباناً (على الكومبيوتر IBM) باستخدام خاصية القوس الأيمن RARC والقوس الأيسر LARC بالتتابع . ولتطويع البرنامج على أى كومبيوتر آخر يمكن بناء البرامج الفرعية المناظرة للأوامر LARC و RARC بالطرق التى تعلمناها أو استخدام هذه البرامج وهى فى الملحق (١) .

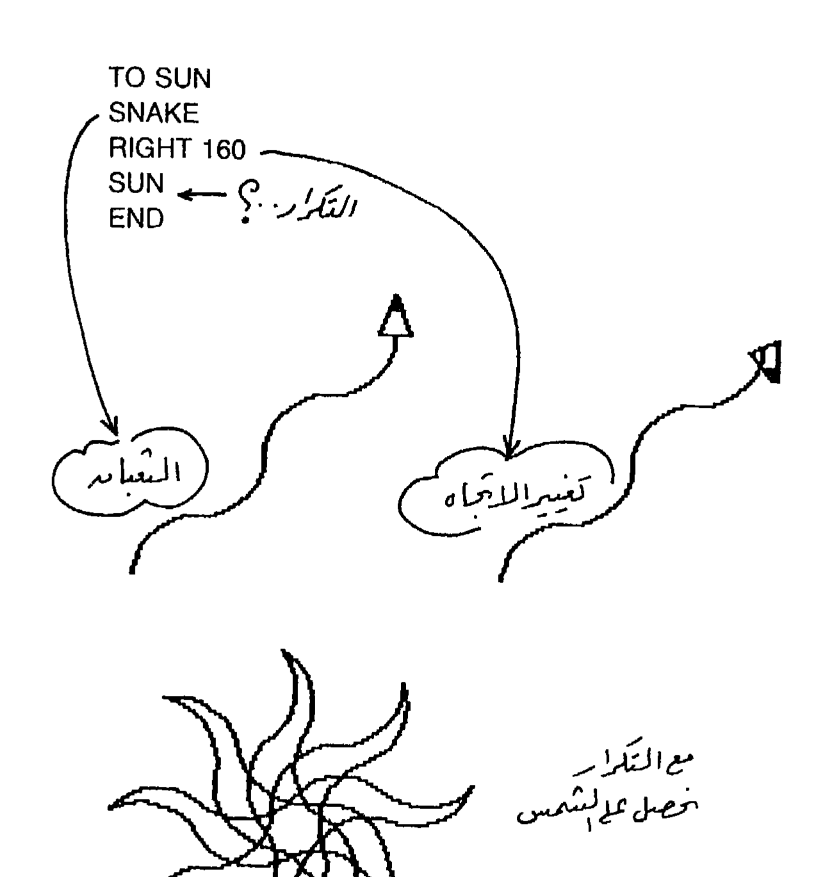
```
TO SNAKE
RARC 30
LARC 30
RARC 30
LARC 30
END
```



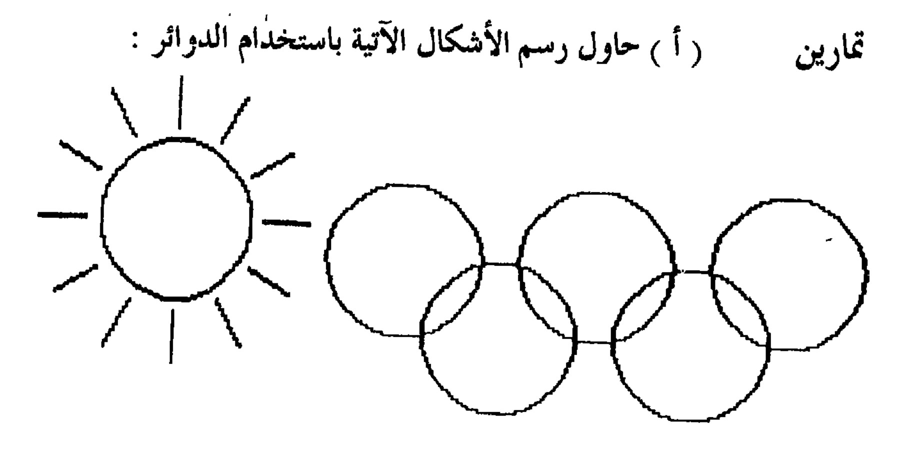
الثعبان وصورته

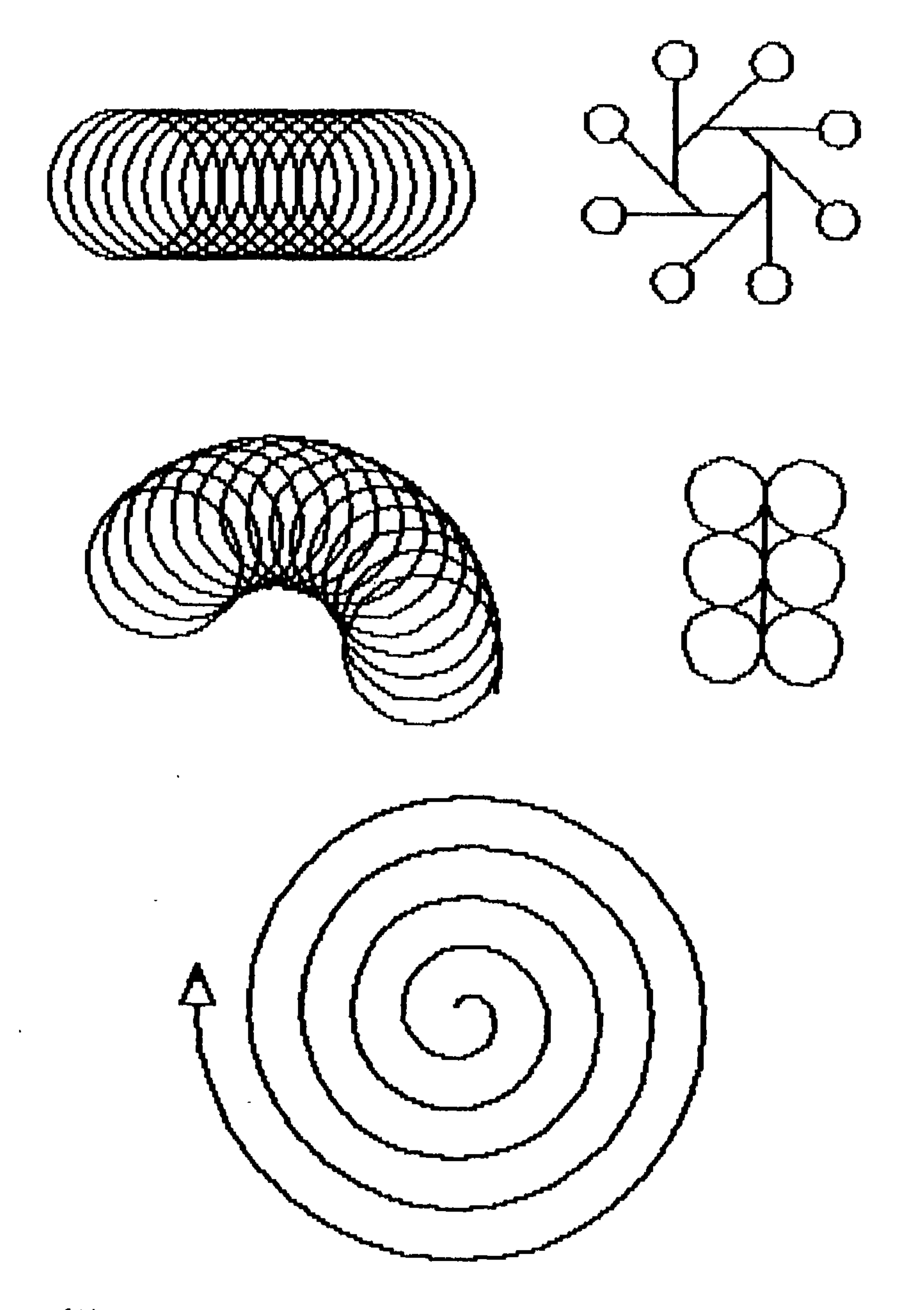
والشكل يوضح الثعبان (أ) الناتج من البرنامج وصورته (ب) التى يمكن برمجتها بتبديل الانحرافات إلى اليمين وإلى اليسار .

ويمكن استغلال هذا البرنامج لرسم الشمس الموضحة فى الشكل التالى باستخدامه كبرنامج فرعى . وسوف يتوقف بعض القراء أقوياء الملاحظة عند الأمر الأخير فى البرنامج الذى يحمل نفس اسم البرنامج SUN .. هل هذا ممكن ؟ إن هذه الطريقة تتميز بها لغة لوجو للتكرار وسوف نتحدث عنها تفصيلاً فى الأبواب القادمة .

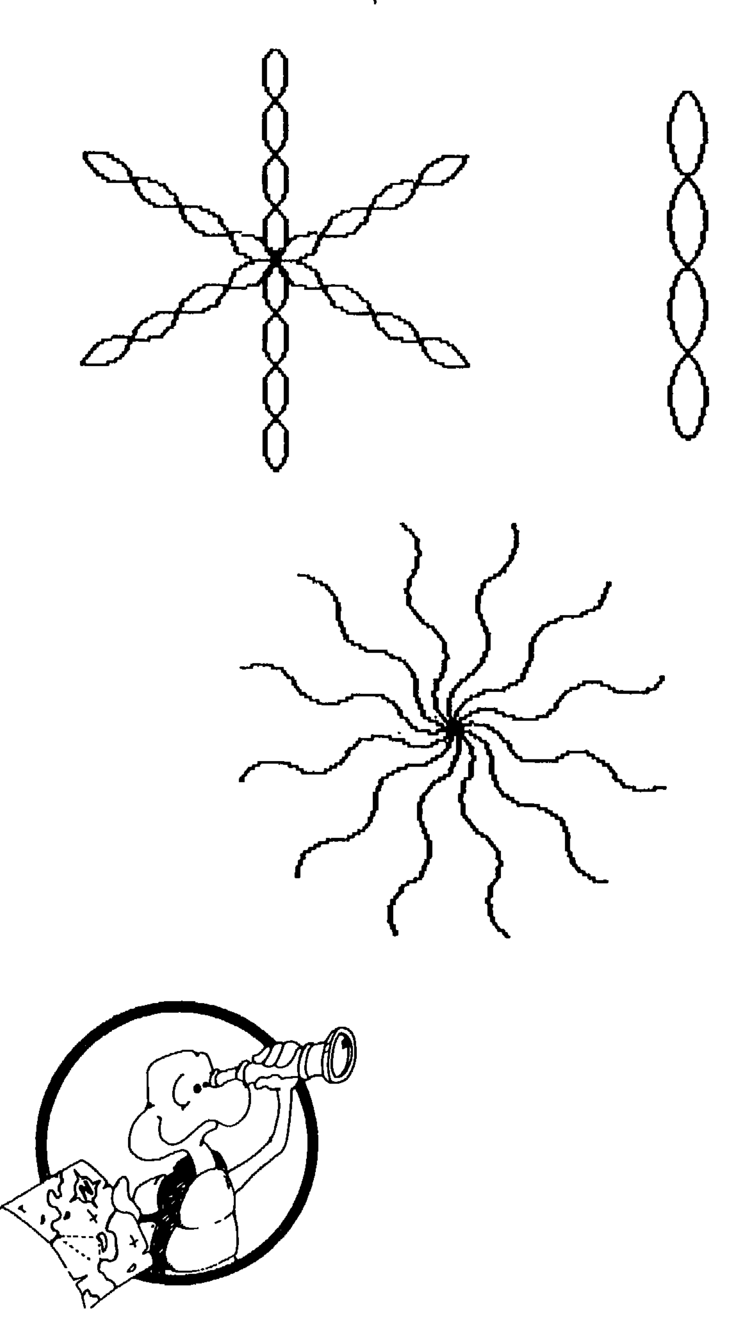


تمارين (أ) حاول رسم الأشكال الآتية باستخدام الدوائر :





(ب) وحاول فى الأشكال الآتية باستخدام المنحنيات :



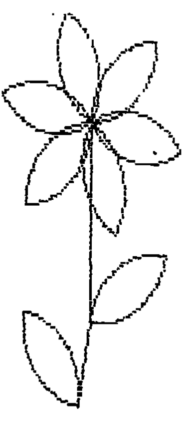
● تطبيقـات :

[١] حدائق الزهور :



عرفنا فى الأبواب السابقة كيف نرسم ورقة الشجرة أو بتلّة الزهرة.ومن البتلّات يمكن تجميع زهرة كاملة باستخدام وسائل التكرار المناسبة .

والبرنامج التالى ARC3 يرسم غصن الشجرة الموضح معه باستخدام البرنامج الفرعى ARC1 الذى يرسم بتلّة واحدة (أو ورقة شجر .. لا فرق بالنسبة للكومبيوتر) .



البرنامج

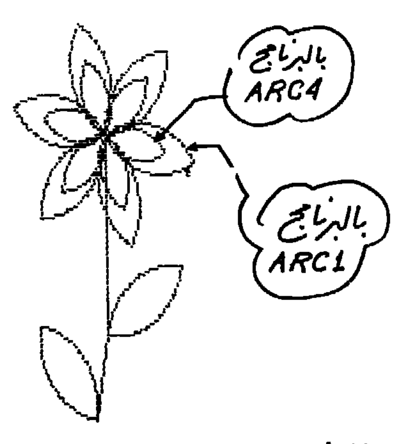
```
TO ARC3
CS HT FD 30
REPEAT 6 [ARC1 LT 60]
BK 70 RT 10 ARC1
BK 30 LT 90 ARC1
END
```

البرنامج الفرعى

```
TO ARC1
REPEAT 45 [FD 1 RT 2]
RT 90
REPEAT 45 [FD 1 RT 2]
RT 90
END
```

وسواء كنت تستخدم البرامج الجاهزة للأقواس والدوائر أو كنت تصمّمها بنفسك فإنه يمكن التحكم فى مساحة الورق أو البتلّة بالتحكم فى زواية انحناء القوس .

والبرنامج الآتى بعد هو تطوير للبرنامج السابق حيث يرسم الزهرة المركبة باستخدام برنامج فرعى جديد ARC 4 علاوة على البرنامج الفرعى السابق ARC1 أما اسم البرنامج الرئيسى فهو FLOWER .



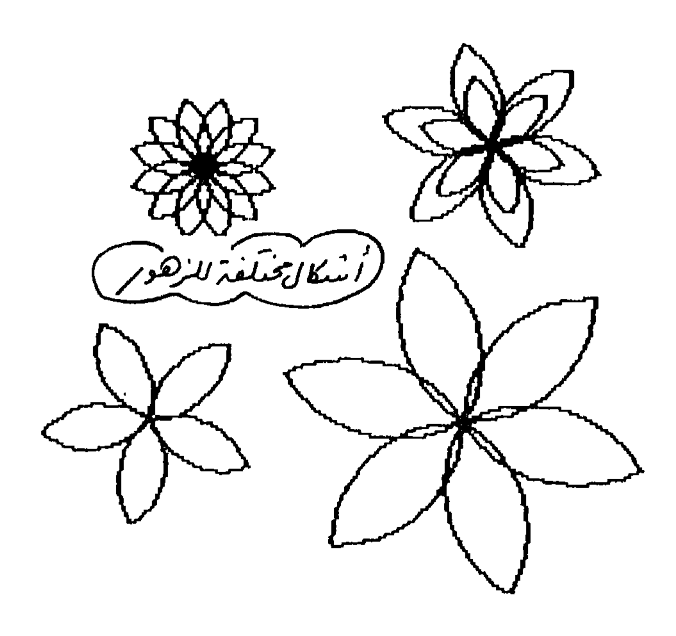
البرنامج

```
TO FLOWER
CS HT FD 30
REPEAT 6 [ARC1 ARC4 LT 60]
BK 70 RT 10 ARC1
BK 30 LT 90 ARC1
END
```

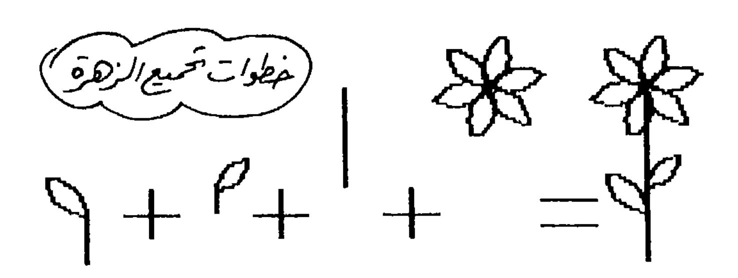
البرنامج الفرعى ARC4

```
TO ARC4
REPEAT 90/3 [FD 1 RT 3]
RT 90
REPEAT 90/3 [FD 1 RT 3]
RT 90
END
```

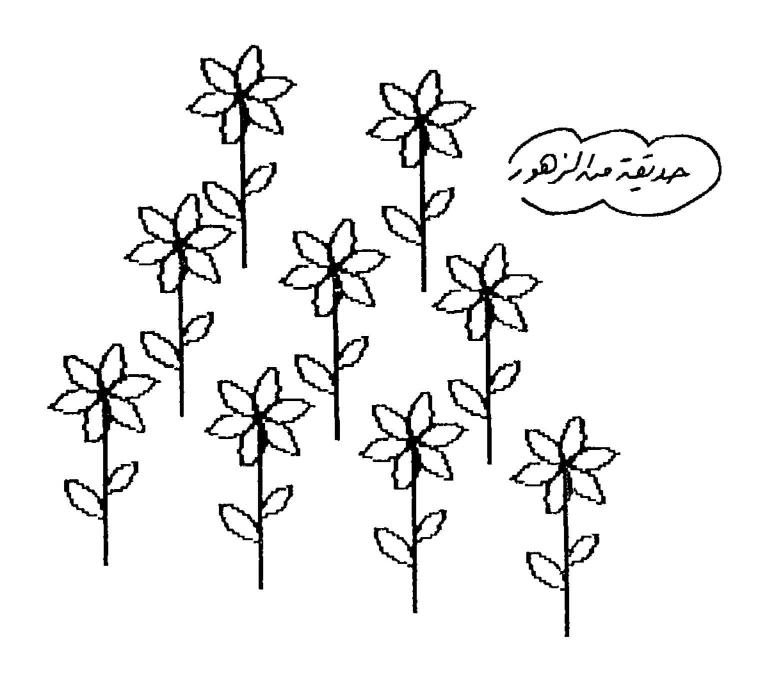
ولا نهاية للأشكال التى يمكن تصميمها باستخدام المنحنيات من الزهور والأوراق والتى تستخدم كوحدة بناء لمجموعات متكاملة من الزهور .



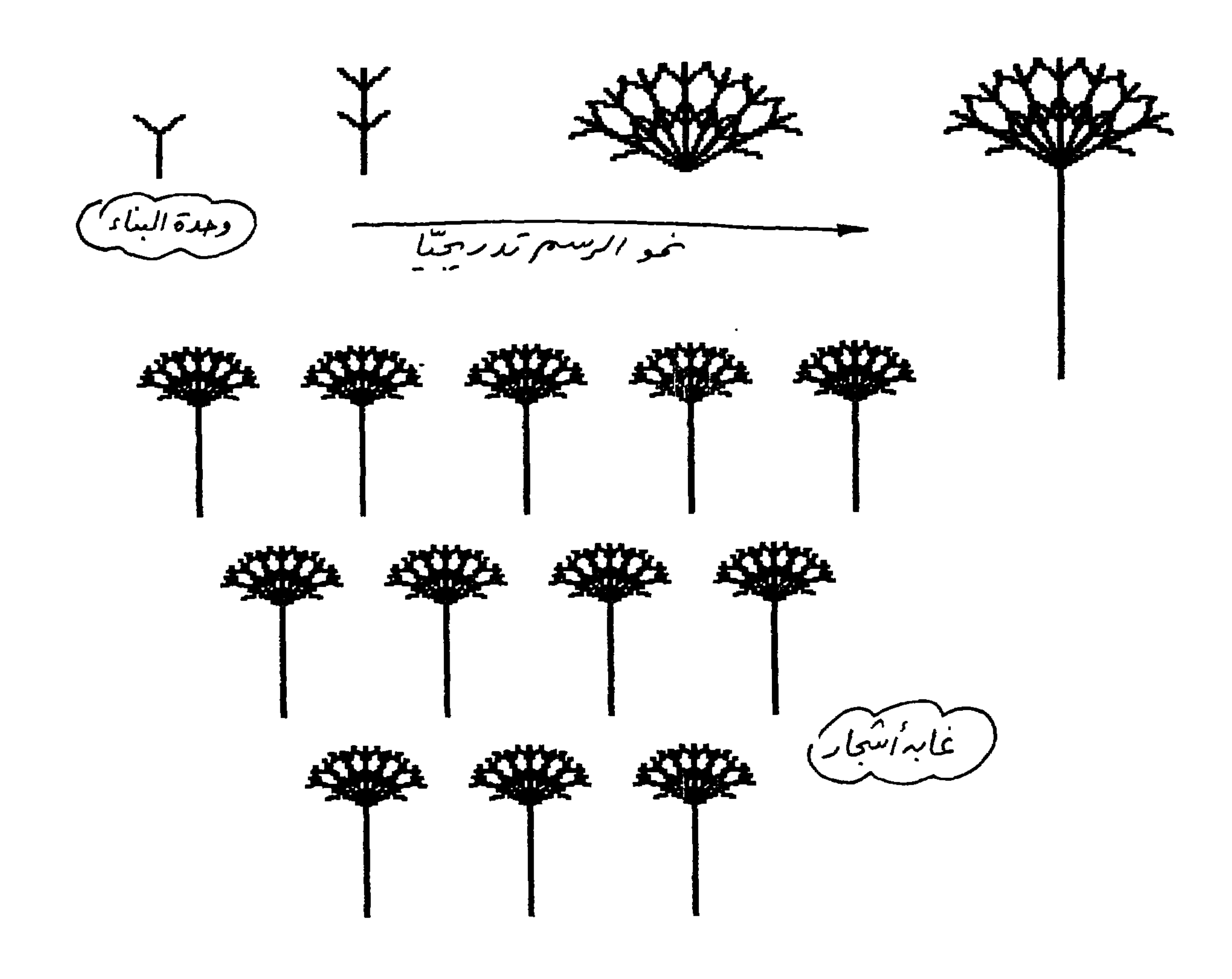
ومن الممكن تقسيم الزهرة إلى عدة أجزاء حتى يسهل رسمها . فمثلاً الزهرة الموضحة يمكن تجميعها من الأشكال المجاورة لها والمكونة من الزهرة السداسية التى رسمناها من قبل والساق (الخط المستقيم) والورقة المائلة إلى اليمين والأخرى المائلة إلى اليسار وهذه هى فكرة البرنامج السابق .



وبناء أول زهرة يمكن تكرار الرسم على مسافات عشوائية أو منتظمة للحصول على الحديقة كاملة .



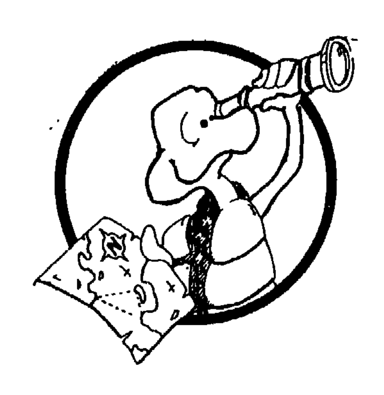
أما الشكل التالى فيوضح كيفية بناء غابة من الأشجار الكثيفة باستخدام وحدة بناء صغيرة لا تحتاج إلى أى مجهود فى البرمجة :

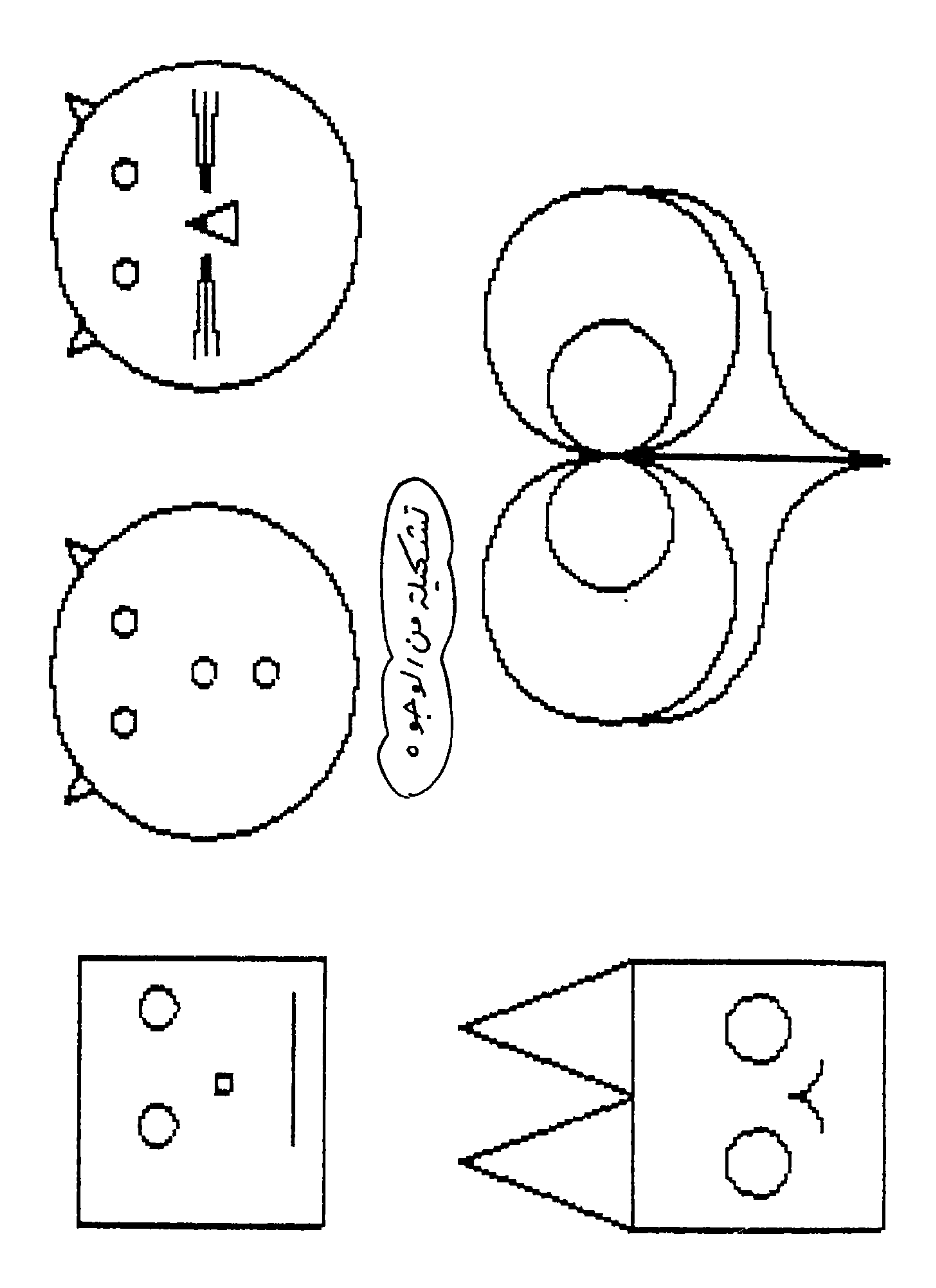

وحدة البناء
نمو الرسم تدريجيا
غابة أشجار

٢ الملامح والوجوه :

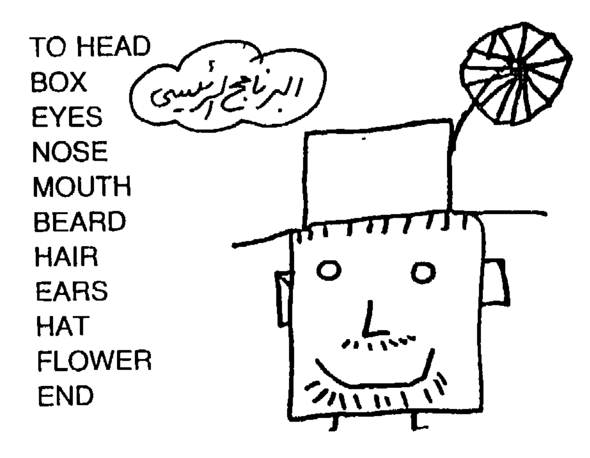
فيما يلى لن نتعرض لتفصيلات البرامج ولكننا نعرض بعض الأفكار عن تكوين الملامح والوجوه بأبسط الطرق فالأشكال الآتية بعد هى أشكال بسيطة نستخدم فيها الدوائر أو المربعات والدوائر معاً . وأهم ما يميّزها التماثل . وفى هذه الحالة تفيدنا خاصية « الصورة على المرآه » فى إتمام الرسم بمجهود أقل فى البرمجة . كما نلاحظ أن السلحفاه قد ظهرت فى أحد الأشكال لترسم شكل أنف القطة ذات الشارب .

حاول رسم هذه الأشكال كتدريب على استخدام الدوائر والمربعات .




تشكيلة من الوجوه

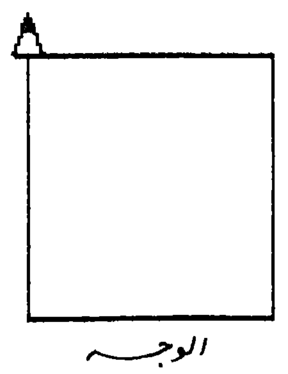
وهذه فكرة برنامج متكامل لرسم الوجه البشرى الموضح فى « الاسكتش » التالى :

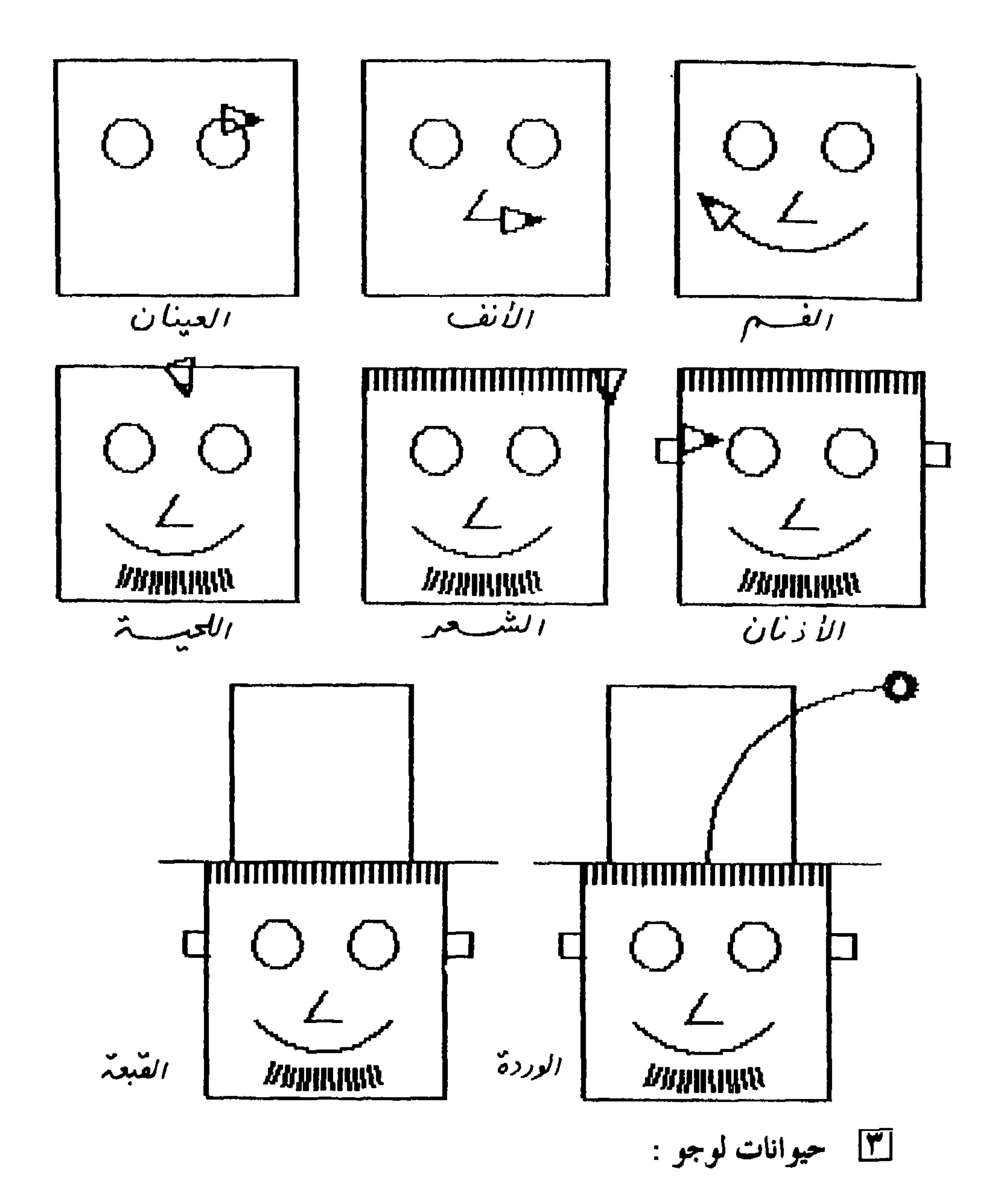


وكما نرى أنه يمكن تقسيم البرنامج الرئيسى إلى عدة أجزاء أو برامج فرعية .

فالوجه نعبر عنه بالمربع (البرنامج BOX) والعينان يتم رسمهما بالدوائر بالبرنامج (EYES) أما الأنف فهى خطّان (NOSE) والفم عبارة عن قوس واحد ، لأعلى إذا كان الرجل مبتسماً ولأسفل إذا كان معكرّ المزاج (MOUTH) .

يلى ذلك اللحية (BEARD) والشعر (HAIR) وهما خطوط مستقيمة والأدنان وهما مربعان صغيران (EARS) ثم القبعة (HAT) وهى مربع أيضاً ، والوردة (FLOWER) وهى عبارة عن قوس ينتهى إلى شكل من الأشكال التى رسمناها من قبل .

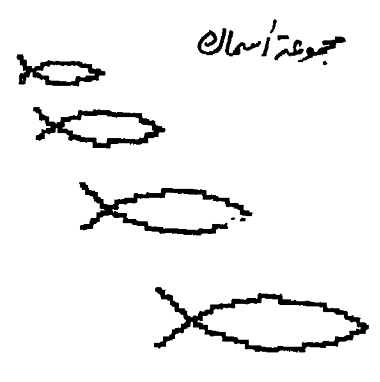

الوجـــه



٣ حيوانات لوجو :

وباستخدام وحدات بناء بسيطة أيضاً يمكن رسم نوعيات مختلفة من الحيوانات الموضحة فى الشكل التالى بعد . فالأسماك عبارة عن أزواج من الأقواس المتقاطعة . والفراشة تبنى من الخطوط والمثلثات . أما القطة والفأر

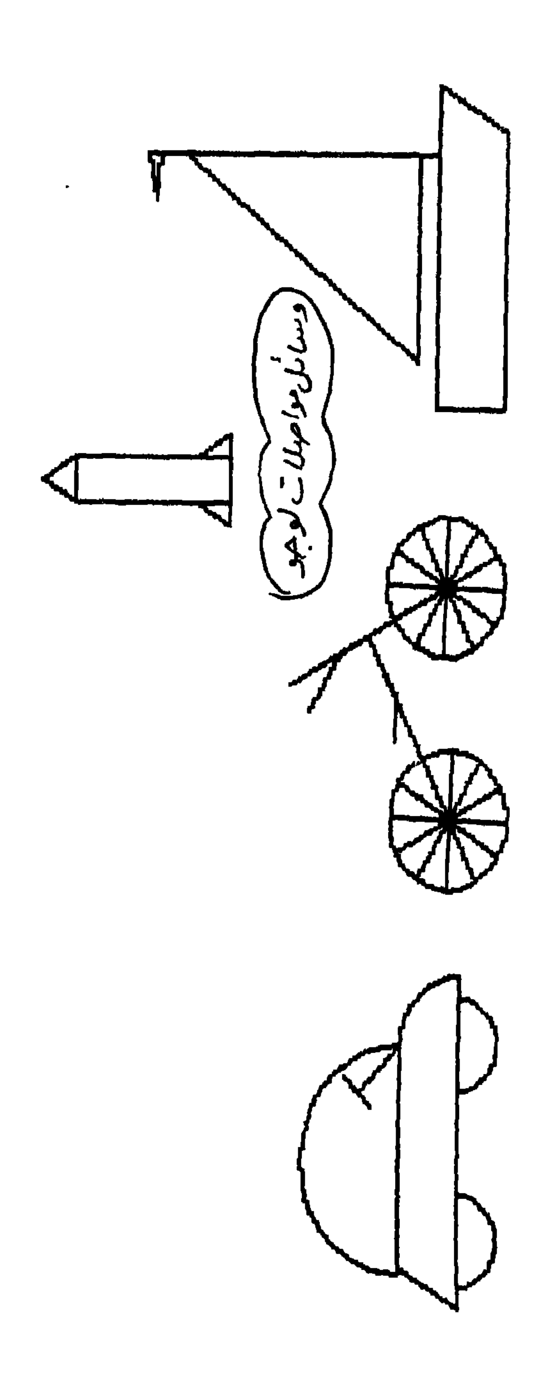
فيبنيان من الدوائر والأقواسِ بصفة أساسية .





٤ ووسائل المواصلات أيضاً :

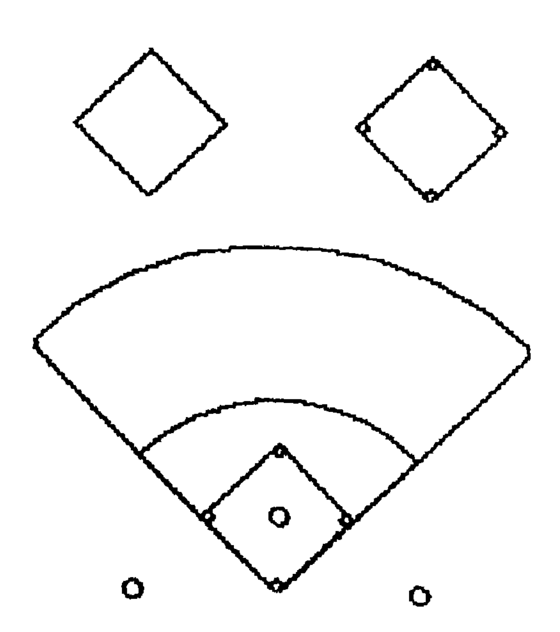
وهذه بعض نماذج من وسائل المواصلات يمكن تصميمها بأبسط الطرق وتشمل المركب الشراعية والسيارة والعجلة والصاروخ .

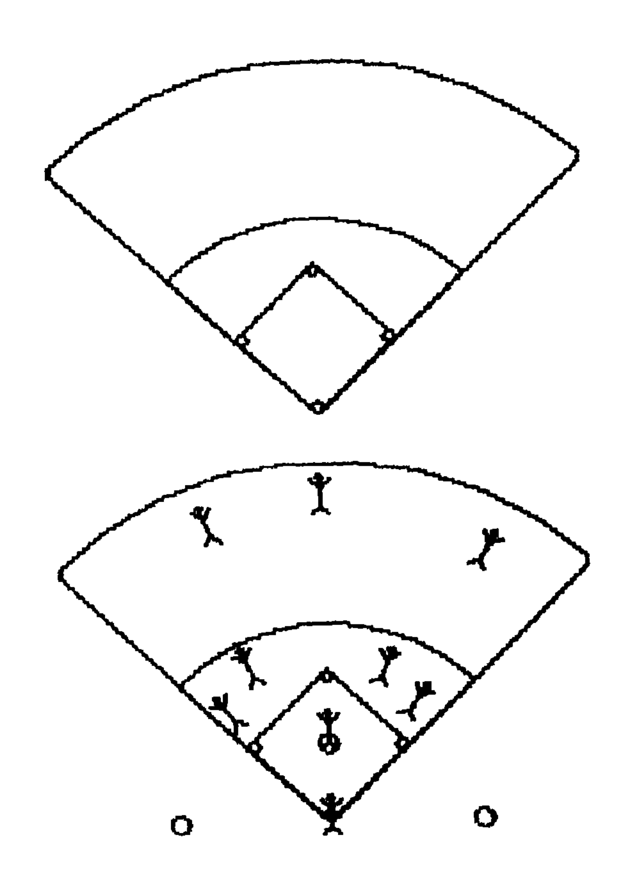

وسائل مواصلات

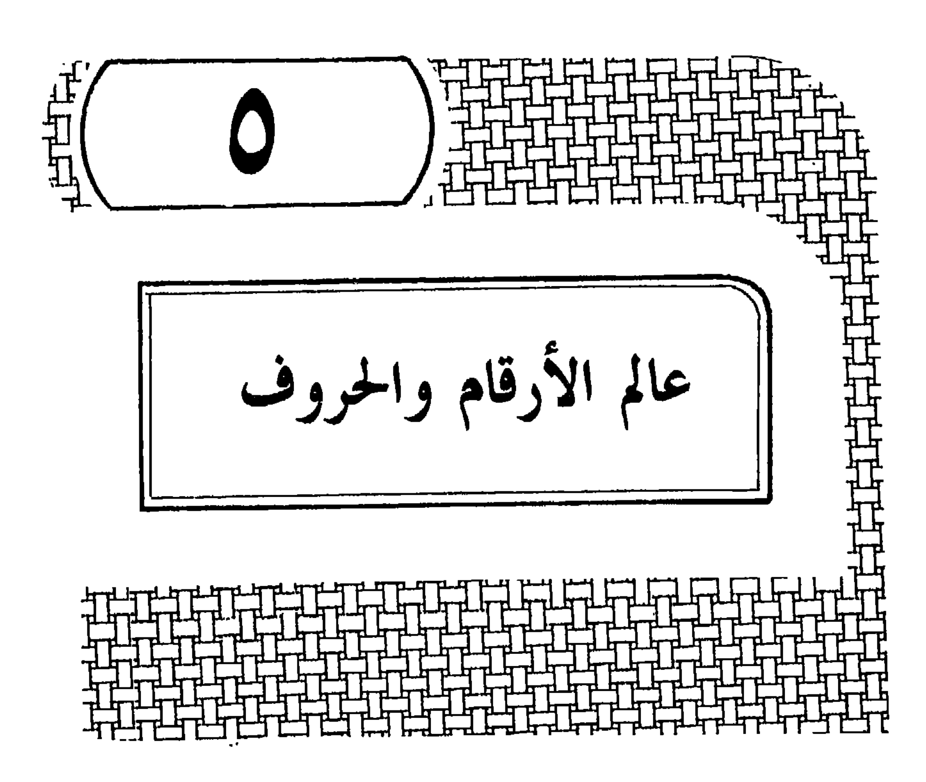
٥ مشروع لعبة كومبيوترية :

وهذه خطوات لرسم ملعب « البيسبول » حيث يبدأ المشروع بمربع واحد ثم ينتهى إلى شكل الملعب المتكامل وفيه اللاعبون .

ولعل بعض الأصدقاء يستطيعون التحول من بالمشروع من مجرد الرسم إلى بناء اللعبة الكومبيوترية الكاملة بعد التمكن من لغة لوجو .







٥

عالم الأرقام والحروف

● شاشة الكتابة TEXTSCREEN

تعرضنا بسرعة من قبل للأمر TEXTSCREEN أو بالصورة المختصرة TS وعلمنا أن هذا الأمر يمسح ما على الشاشة من كتابة أو رسم ويضع علامة الاستعداد (؟) يسار أعلى الشاشة ، وهو مكان مناسب لها عندما نريد أن نكتب أو نحسب بعض الحسابات بلغة لوجو .

● البيانات فى لغة لوجو :



البيانات التى يتعامل معها الكومبيوتر من خلال لغة لوجو ثلاثة أنواع :

١ ــ الأعداد (Numbers)

وهو يستطيع مع الأعداد أن يؤدى مختلف العمليات الحسابية والمنطقية كما سنرى .

٢ ــ الكلمات (WORDS)

أما الكلمات فهى التراكيب المختلفة لعناصر اللغة مثل الحروف الأبجدية والأرقام والعلامات الخاصة .

٣ ــ القوائم (LISTS)

والقوائم هى تركيبة من مجموعة كلمات .

وسوف نرى فى الفقرة القادمة كيف يتعامل الكومبيوتر مع كل نوع من أنواع هذه البيانات .

(أ)

•• التعامل مع الأعداد فى لغة لوجو ••

• العمليات الحاسبية :

الأرقام هى أحب شيء إلى الكومبيوتر . لأنه يرى العالم الخارجى فى شكل أرقام . ولو أنك أعطيته بياناً ما مكوناً من الحروف الأبجدية مثل اسم شخص ، فإنه فى الحقيقة يخزّن فى صورة أرقام . لكنك عندما تطلب منه طبعه على الشاشة فإنه يحوّله مرة أخرى إلى حروف حتى نفهم نحن !

والآن نرى معاً كيف يمكن إجراء بعض الحسابات على الكومبيوتر باستخدام لغة لوجو ولنبدأ **بعملية الجمع** :

? PRINT 4 + 5 ← أمر الجمع
9 ← النتيجة
? ← علامة الاستعداد

إذن فعملية الجمع تتم باستخدام الأمر PRINT . والرمز المستخدم فى عملية الجمع هو (+) . ويراعى ترك مسافة بين الكلمة PRINT والأرقام التى تليها .

كذلك فإن الرمز المستخدم فى عملية الطرح هو (−) وفى الضرب هو (*) وفى القسمة هو (/) .

ولنجرب الأمثلة الآتية :

```
PRINT 4 + 5
PRINT 36 - 6
PRINT 36 / 7
PRINT (3 + 5) * 7
```

وفى المثال الأخير نلاحظ ظهور الأقواس مع الأعداد . لماذا ؟

إن هذا القوس يخبر الكومبيوتر أن يجمع العددين 5 + 3 أولاً ثم يضرب

الناتج فى العدد 7 .

ولو كانت العملية الأولى عملية طرح لزم استخدام الأقواس أيضاً . ولكن إذا كانت العملية الأولى عملية ضرب أو قسمة فيمكن الاستغناء عن الأقواس لأن الكومبيوتر ينفذ الضرب والقسمة قبل الجمع والطرح عادة .

وللتأكد من ذلك جرب الآتى :

```
PRINT  3 + 5 * 7
PRINT  (3 + 5) * 7
```

الناتج الذى نحصل عليه من العملية الأولى هو 38 .
والناتج الذى نحصل عليه من العملية الثانية هو 56 .

أى أن الكومبيوتر فى العملية الأولى قد أنجز عملية الضرب قبل عملية الجمع وفى العملية الثانية أنجز عملية الجمع قبل عملية الضرب .

● المقارنات TRUE, FALSE

يمكن للكومبيوتر بجانب العملية الحاسبية أن يتخذ قراراً منطقياً أيضاً . والقرار المنطقى يؤدى إلى أحد نتيجتين : « صحيح » (TRUE) أو « خطأ » (FALSE) .

والقرار المنطقى يكون عادة إجابة لسؤال نسأله نحن للكومبيوتر مثل :

```
PRINT  5 = 8
```

هذا معناه : « هل الرقم 5 يساوى الرقم 8 » ؟

ولأن الإجابة هى « لا » فإن الكومبيوتر يطبع على الشاشة كلمة FALSE . فلنسأله الآن هذا السؤال :

```
PRINT  5 < 8
```

هذا معناه : « هل 5 أصغر من 8 » ؟

الإجابة هى TRUE بمعنى نعم .

جرب الآن السؤال الآتى :

```
PRINT  5 > 8
```

وهذا معناه : « هل 5 أكبر من 8 » ؟

الإجابة المنتظرة هى FALSE بمعنى لا .

ومن الجائز أيضاً أن يشمل السؤال عملية حسابية مع القرار المنطقى المطلوب مثل :

```
PRINT   5 = (8 - 3)
```

وهذا معناه : « هل 5 مساو لناتج عملية الطرح (3 - 8) ؟

والإجابة هى TRUE .

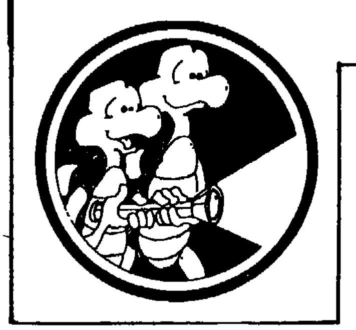
فلاش

هل نحتاج إلى الأقواس فى المثال السابق ؟

فى الحقيقة لا . لأن الكومبيوتر يؤدى أية عملية حسابية قبل المقارنات . لذلك فهو يحسب أولاً المقدار (3 - 8) بصرف النظر عن وجود الأقواس ثم يقارن العدد 5 بالنتيجة .

وجرّب الآتى للتأكد :

```
PRINT   5 = 8 - 3
PRINT   5 = (8 - 3)
```



● مثـال :

هذه بعض العمليات الحسابية على الكومبيوتر المنزلى سنكلير نقدمها كمثال عام لما سبق . ولعلنا نلاحظ أن لغة لوجو للكومبيوتر IBM هى هى للكومبيوتر سنكلير أو غيره ، فى حدود التطبيقات التى نعرضها فى هذا الكتاب ومع ذلك فسوف نشير إلى الأجزاء التى قد يختص بها كومبيوتر IBM إذا تعرضنا لها :

ونلاحظ فى هذا المثال استخدام الأمر المختصر PR بدلاً من PRINT .

كومبيوتر سنكلير

```
WELCOME TO SINCLAIR LOGO
© SOLI/LCSI 1984 VER. 1.5
?pr 5+8
13
?pr 5*5
25
?pr 55>6
TRUE
?pr 6<5
FALSE
?pr 6=26-20
TRUE
?pr 6=(26-20)
TRUE
?
```

عمليات حسابية (للأسطر الأربعة الأولى)

مقارنات (للأسطر الباقية)

● هناك دائماً طريقة أخرى :

يمكن أيضاً أداء العمليات الحسابية بطريقة مختلفة بدون استخدام المؤثرات (+ * /) وهذه بعض أمثلة :

* عملية الضرب :

PR PRODUCT 4 4 ← الأمر

16 ← الإجابة

* عملية القسمة :

PR DIV 7 4

3.5

* عملية الجمع :

```
PR SUM 50 40
90
```

● عملية المقارنة (=) :

الأمر PR EQUALP 5 7

الإجابة FALSE

ملاحظة

هل لاحظت ظهر العلامة العشرية (.) فى مثال القسمة . إن لوجو تتعامل مع الأعداد الصحيحة والأعداد الحقيقية (التى تحتوى على كسور) وفى الفقرة القادمة نقدم بعض وسائل التعامل مع الكسور بالتقريب أو الحذف .

● التقريب (ROUND)

اكتب الأمر التالى على الشاشة وشاهد النتيجة :

```
PRINT ROUND 5.463
```

نتيجة هذا الأمر هى طبع العدد 5 .

جرب الآن الأمر التالى :

```
PRINT ROUND 5.964
```

نتيجة هذا الأمر هى العدد 6 .

أى أن عملية التقريب قد أجريت على الكسر لأقرب رقم صحيح .

وهذه هى إجابات الكومبيوتر سنكلير :

```
?print round 5.463
5
?print round 5.964
6
```

● الحذف INT

أما هذه العملية فينتج عنها حذف الجزء الكسرى من العدد الحقيقى بصرف النظر عن قيمة الكسر .

أنظر هذه الأمثلة :

```
? PR  INT  5.2129
  5
? PR  INT  5.5
  5
? PR  INT  -5.5
  -5
```

● الأعداد العشوائية RANDOM

أما العملية RANDOM فهى تشبه النرد (زهر الطاولة) بمعنى أنها تنتج أى رقم عشوائى تماماً كما تلقى النرد فتظهر على سطحه المواجه لك أية أرقام عشوائية .

وحدود الرقم العشوائى يتحكم فيها العدد المُدخل مع كلمة RANDOM كالمثال الآتى :

? PR RANDOM 6

هذا الأمر يطبع عدداً عشوائياً يتراوح بين صفر وخمسة .

وأيضاً :

? PRINT RANDOM 50

هذا الأمر يطبع على الشاشة عدداً عشوائياً بين صفر و ٤٩ .

أى أن الحد الأقصى للعدد الذى تنتجه كلمة RANDOM أقل من الرقم المُدخل بعدها بمقدار 1 .

لذلك يمكن كتابة الكلمة RANDOM فى صورة عامة كالآتى :

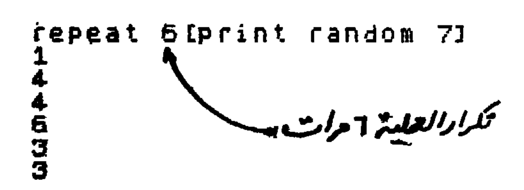
RANDOM n

ويكون أقصى عدد تنتجه كلمة RANDOM هو (n − 1) .

● مثـال :

يمكن هنا استخدام أمر التكرار REPEAT لإنتاج عدد من الأرقام العشوائية التى نستطيع استغلالها فيما بعد فى الكثير من التطبيقات لا سيما فى الرسم .

وهذا المثال منفذ على الكومبيوتر المنزلى سنكلير :



الأرقام المطبوعة هنا تتراوح بين الصفر والستة .

تجربة

هل تستطيع إجراء تعديل على المثال السابق بحيث تحاكى النتيجة عملية رمى النرد أى تطبع أرقاماً تتراوح بين الواحد والستة .

● عالم الأعداد فى لغة لوجو :

إن عالم الأعداد فى لغة لوجو عالم متسع . وهى تقدم إمكانات كثيرة جداً للتعامل مع الأعداد ربما لا تجود بها لغة أخرى من لغات الكومبيوتر .

ولا نعتقد أن مستوى هذا الكتاب يكفى لأن نقول كل شيء عن إمكانات التعامل مع الأعداد ، فبعض هذه الإمكانات لا يهم إلاّ الرياضيين ومع ذلك فاعتقادى أن إشارة سريعة لمثل هذه التطبيقات تكفى لإرشادهم إلى الطريق .

وهذه بعض العمليات الحسابية والرياضية التى تتضمنها لغة لوجو :



● النسـب المثلثيـة :

تحتوى لغة لوجو على النسب المثلثية الآتية :

حا	SINE n أو SIN n
جتا	COSINE n أو COS n
ظا	TANGENT n أو TAN n

● مثـال :

```
? PR TAN 50
1.1917536
```

● النسب المثلثية العكسية :

جا-١	ARCSIN n
جتا-١	ARCCOS n
ظا-١	ARCTAN n

● مثـال :

? PR ARCTAN 1 ← الأمر

45 ← الإجابة

● مقلوبات النسب المثلثية :

ظتا	COTANGENT n أو COT n

● مثـال :

? PR COT 45 ← الأمر

1 ← الإجابة

● باقى القسمة REMAINDER a b

تعطى هذه العملية باقى قسمة العدد a على العدد b كالأمثلة الآتية :

```
? PR   REMAINDER   16   4
0           ←
? PR   REMAINDER   16   5
  1
```

اطبع باقى قسمة ١٦ على ٤

الإجابة .. صفراً

وفى هذا المثال نرى أن باقى قسمة 16 على 5 هو العدد 1 .

● **إيجاد الجذر التربيعى** SQRT n

● **مثال :**

```
? PR   SQRT   49
  7

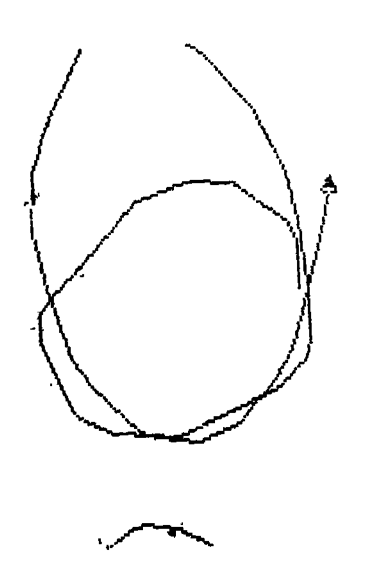
? PR   SQRT   4567
  67.596
```

● **مثال عام** **الرسم فى إحداثيات عشوائية :**

لو أنك أدخلت هذا الأمر إلى الكومبيوتر :

```
? REPEAT   50 [ FD RANDOM 30 LT RANDOM  30 ]
```

فسوف تشاهد السلحفاه تتحرك فى اتجاهات عشوائية فتحصل على رسومات عشوائية على الشاشة بمعنى أنها يختلف شكلها فى كل مرة يُنفذ فيها هذا الأمر . والشكل التالى هو أحد الأشكال العشوائية التى يمكن مشاهدتها على الشاشة .



أحد أشكال
التنفيذ العشوائية للرسم

(ب)
●● التعامل مع الكلمات والقوائم فى لغة لوجو ●●

● ما هى كلمات لوجو ؟ LOGO WORDS

تتكون أى لغة من الحروف والأرقام والعلامات الخاصة الموجودة على لوحة الأزرار . وهناك اسم عام نطلقه على أى من هذه المكونات وهو « **اللبنة** » **(character)** واللبنة معناها أصغر جزء فى بناء اللغة (أو الحجر الذى يُبنى منه المنزل) .

والكلمة فى لغة لوجو هى تركيبة من هذه اللبنات ولا يشترط أن تحمل معنى ما . وعادة تسبق كلمة لوجو علامة الاقتباس المزدوجة (") (ولكنها لا توضع فى مؤخرتها مثل اللغات الأخرى) .

ومع ذلك فهناك بعض اللبنات لا يجوز على أن تحتوى عليها الكلمة مثل :

+ , - , * , / , (,) , [,]

ولنجرب معاً هذه الأوامر ونشاهد رد فعل لوجو :

```
PRINT "HELLO
PRINT "ABCXYZ
PRINT "R2D2
PRINT "AB.$ -) * *
PRINT "3 + 4
```

وتعتبر الأرقام أيضاً **كلمات** (WORDS) فى لغة لوجو وقد تستخدم مع علامة الاقتباس أو بدونها كالمثال الآتى :

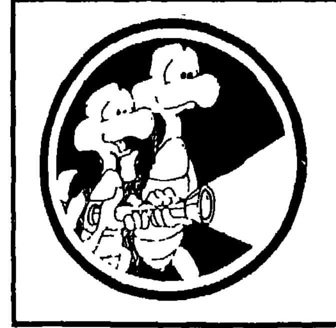
```
PRINT "25
PRINT "25 + "25
```

فى الأمر الأول يطبع الكومبيوتر العدد 25 .

وفى الأمر الثانى يطبع العدد 50 كمجموع لكل من 25 ، 25 .

وهذا يكافىء تماماً الأمر : PRINT 25 + 25

ملاحظة



هناك فرق بين الحرفيات (strings) فى لغة بيسك وبين الكلمات فى لغة لوجو ولذلك يجب الحذر من الخلط بينهما بالنسبة لمن تعلّم لغة بيسك من قبل .

ويجوز أن تكتب العلامة (") بعد الأمر PRINT دون أن يليها شيء ما . وهذا بمثابة الأمر « اترك سطراً خالياً » أى :

? PRINT

● تحديد نهاية الكلمة :

وتنتهى كلمة لوجو عند أول مسافة خالية . **فالكلمة لا يجوز أن تحتوى على مسافة خالية** ، تماماً مثل كلمات اللغة العربية أو الإنجليزية .

أدخل هذا الأمر للكومبيوتر وشاهد الرد على الشاشة :

? PRINT " my name is SALLY

فى هذه الحالة سوف يعتبر الكومبيوتر أن (my) هى الكلمة المقصودة أما ما بعدها فسوف يشكو من عدم الفهم له لذلك يكون رد الفعل كالآتى :

? PRINT " my name is SALLY ←	الأمر
my ← الإجابة	
I don't Know how to NAME ←	الشكوى
? ← علامة استعداد جديدة	

إذن فلنجرب إعطاء هذا الأمر الجديد حيث نضع علامة اقتباس أمام كل كلمة :

? PRINT "my " name " is "SALLY ←	الأمر
my ←	الإجابة
You don't say what to do with name	
?	شكوى جديدة

آه لقد عرف أن الكلمة التالية لكلمة my هى أيضاً كلمة ولكنه توقف عندها أيضاً وأرسل لنا رسالة بالخطأ تحمل الشكوى : « لم تخبرنى ماذا أفعل بكلمة name » .

وبالطبع فإنه لم يقرأ بقية الجملة . هناك حل وسوف نعرفه فى حينه .

● القوائم فى لغة لوجو LISTS

أما القوائم فهى تركيبة من كلمات لوجو .

ونضع القائمة عادة بين قوسين مربعين [ ] ولنكتب هذه الأمثلة ونشاهد النتيجة على الشاشة :

```
PRINT [HELLO THERE]
PRINT [1 2 3 4 5 6]
PRINT [MY NAME IS DAN]
PRINT [THIS IS A LIST: [THIS IS A LIST: ]]
PRINT [[A B] [C D] [E F]]
```

ولعلنا نلاحظ أنه بداخل القوسين المربعين يمكننا أن نستخدم أى لبنة من اللبنات الموجودة على لوحة الأزرار حتى لو كان القوس المربع نفسه . فأمر الطباعة هنا ينسخ كل ما بداخل القوسين ويطبعه على الشاشة . والقوائم فى لغة لوجو تناظر الحرفيات strings فى لغة بيسك .

ويجوز أن نترك القوسين المربعين فارغين وهذا معناه ترك سطر خال أيضاً .

```
PRINT [ ]
```

● توصيل كلمتين معاً WORD

هناك أوامر خاصة بالكلمات والقوائم يمكننا من خلالها أن نكوّن التراكيب المختلفة فمثلاً الأمر WORD يقوم **بوصل كلمتين معاً كالآتى :**

```
PRINT WORD "BIG "WORD
PRINT WORD "WO "RD
```

فى المثال الأول يقوم الكومبيوتر بطبع كلمة BIGWORD .
وفى المثال الثانى يقوم بطبع كلمة WORD .

ولكن لا يجوز توصيل أكثر من كلمتين بهذه الطريقة ، ولو أنك حاولت فسوف يتوقف الكومبيوتر عند الكلمة الثالثة ويرسل لك الشكوى المعروفة : « ماذا أفعل بهذه الكلمة الثالثة .. » .

● توصيل أكثر من كلمتين (WORD "...

يمكن للأمر WORD أن يقبل أكثر من كلمتين بشرط استخدام الأقواس العادية ( ) لتشمل كل الكلمات المُدخلة كالمثال الآتى :

```
PRINT (WORD "BIG "GER "WORD )
```

مسافة خالية

هنا سيقوم الكومبيوتر بتوصيل كل ما بين القوسين ويطبع على الشاشة الكلمة BIGGERWORD .

ملاحظة



هناك مصادر شهيرة للخطأ فى الاستخدام السابق للأمر WORD أولها : ضرورة ترك مسافة خالية قبل القوس الأخير "(" وإلاّ سيكتب الكومبيوتر هذا القوس ضمن الكلمة الجديدة ظناً منه أنه جزء من الكلمة الأخيرة (فى أغلب الطرازات) .

أما الخطأ الشائع الثانى فهو وضع كلمة WORD خارج القوسين كالآتى :

```
PRINT WORD ("BIG "GER "WORD )
      ↖── X
```

• توصيل الكلمات والقوائم فى جملة SENTENCE

بهذا الأمر الجديد SENTENCE يمكن توصيل الكلمات والقوائم لتكوين قائمة كبيرة . **ويستخدم هذا الأمر مُدخلين فقط** وقد يكون أى منهما كلمة أو قائمة .

ولنر معاً هذه الأمثلة :

```
PRINT SENTENCE "A [WORD PLUS A LIST]
PRINT SENTENCE [A LIST PLUS A] "WORD
PRINT SENTENCE "TWO "WORDS
```

والأمر SENTENCE يتم اختصاره إلى **SE** كما فى المثال الآتى لتوصيل قائمتين معاً :

```
PRINT SE [TWO LISTS] [MAKE A LIST, TOO]
```

● توصيل العديد من القوائم والكلمات **(SENTENCE..**

وكما مع الأمر WORD فإن الأمر SENTENCE يمكنه أيضاً استخدام أكثر من مُدخلين (INPUT) إذا استخدمنا معه الأقواس العادية ( ) .

● **مثـال :**

```
PRINT (SENTENCE "THIS [WILL BECOME] [ONE LIST] "TOO )
```

● طباعة جزء من كلمة أو قائمة

FIRST, LAST, BUTFIRST, BUTLAST

هاك أربعة أوامر جُدد للتعامل مع الكلمات والقوائم وتقطيعها إلى أجزاء .

● الأمر الأول هو FIRST وهو يستخدم لطباعة الحرف الأول من الكلمة أو الكلمة الأولى من القائمة .

● الأمر الثانى هو LSDY ويستخدم لطباعة آخر حرف فى كلمة أو آخر كلمة فى قائمة .

● الأمر الثالث هو BUTFIRST بمعنى « ما عدا الأول » أى طبع كل الكلمة ما عدا الحرف الأول . أو طبع كل القائمة ما عدا الكلمة الأولى .

● الأمر الرابع هو BUTLAST بمعنى « ما عدا الأخير » أى طبع الكلمة ما عدا الحرف الأخير . أو طبع القائمة ما عدا الكلمة الأخيرة .

فلنر الأمثلة مع الكلمة :

● **مثال** ١ :

```
PRINT FIRST "HELLO
PRINT LAST "HELLO
PRINT BUTFIRST "HELLO
PRINT BUTLAST "HELLO
```

نتيجة تنفيذ المثال هي : طبع الحرف H نتيجة للأمر الأول . يليه طبع الحرف O نتيجة للأمر الثانى على سطر جديد . يليه طبع الجزء ELLO من كلمة HELLO . ثم الجزء HELL على السطر الرابع .

أى أن نتيجة التغير كالآتى :

« أول حرف .. »	**نتيجة الأمر الأول**	**H** ←
« آخر حرف .. »	**نتيجة الأمر الثانى**	**O** ←
« ما عدا الأول .. »	**نتيجة الأمر الثالث**	**HLLO** ←
« ما عدا الأخير .. »	**نتيجة الأمر الرابع.**	**HELL** ←

● **مثال** [٢] :

هنا نستخدم نفس الأوامر ولكن مع القوائم LISTS :

```
PRINT FIRST [HELLO MY FRIEND]
PRINT LAST [HELLO MY FRIEND]
PRINT BUTFIRST [HELLO MY FRIEND]
PRINT BUTLAST [HELLO MY FRIEND]
```

وتكون نتيجة التنفيذ هى :

أول كلمة .. »	**HELLO** ←
« آخر كلمة .. »	**FRIEND** ←
« ما عدا أول كلمة .. »	**MY FRIEND** ←
« ما عدا آخر كلمة .. »	**HELLO MY** ←

● **مثال ٣** :

ومن الممكن إجراء تركيبات مختلفة من هذه الأوامر « لتقطيع » الكلمة أو القائمة لانتقاء جزء معيّن منها كالآتى :

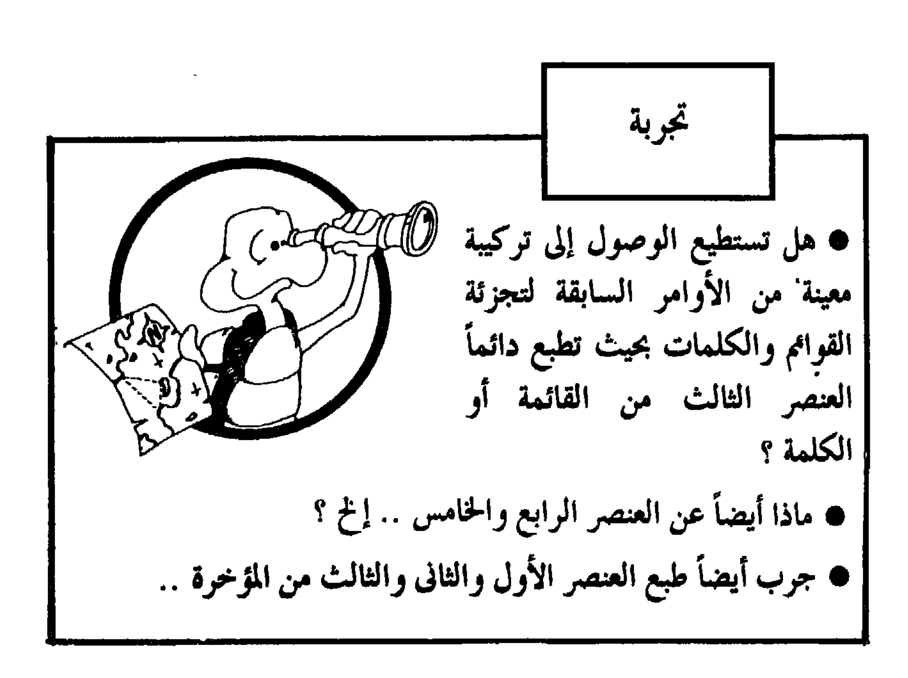
```
PRINT FIRST BUTFIRST [HELLO MY FRIEND]
PRINT BUTFIRST FIRST [HELLO MY FRIEND]
```

- الأمر الأول ينتج عنه طبع كلمة MY لماذا ؟

أن الأمر BUTFIRST سيقوم بطبع القائمة MY FRIEND تاركاً الكلمة الأولى . ثم يأتى الأمر FIRST ليطبع الكلمة الأولى وهى MY .

- أما الأمر الثانى فينتج عنه طبع الكلمة ELLO كالآتى :

الجزء الأول من الأمر وهو FIRST سيطبع كلمة HELLO كاملة وعندما يأتى دور الأمر BUTFIRST سيجد أمامه كلمة واحدة فيطبع منها كل الحروف ما عدا الأول .



تجربة

- **هل تستطيع الوصول إلى تركيبة معينة من الأوامر السابقة لتجزئة القوائم والكلمات بحيث تطبع دائماً العنصر الثالث من القائمة أو الكلمة ؟**
- **ماذا أيضاً عن العنصر الرابع والخامس .. إلخ ؟**
- **جرب أيضاً طبع العنصر الأول والثانى والثالث من المؤخرة ..**

ومن الجدير بالذكر أن بعض الأوامر الأخيرة آيضاً يمكن اختصارها كالآتى :

الاختصار	الأمر
BF BL	BUTFIRST BUTLAST

●● أمثلة عامة ●●

● **الكومبيوتر يسأل وأنت تجيب** **READLIST**

● **مثال** [1] :

اكتب هذا البرنامج الفرعى وشاهد النتيجة :

الكومبيوتر ينتظر هنا..

```
TO TALK
PRINT [PLEASE TYPE SOMETHING FOR ME TO SAY]
PRINT SENTENCE [YOU JUST MADE ME SAY] READLIST
END
```

عند تشغيل هذا البرنامج الفرعى سوف يطبع الكومبيوتر على الشاشة الرسالة التى فى القائمة الأولى وهى :

PLEASE TYPE SOMETHING FOR ME TO SAY

بمعنى اكتب لى شيئاً لأقوله ..

وسوف ينتظر منك كتابة أى شىء على الشاشة ..

فإذا ما كتبت شيئاً وليكن كلمة "hello" ثم أدخلتها بالضغط على الزر "ENTER" فإنه يستقبل هذه الكلمة وعلى الفور يطبع الرسالة الآتية :

YOU JUST MADE ME SAY hello

بمعنى : لقد جعلتنى أقول hello .

والأمر الجديد READLIST هو الذى يجعل الكومبيوتر يتوقف لاستقبال قائمة ما وتخزينها فى القائمة READLIST .

ولذلك فعند تنفيذ الأمر الثانى فى البرنامج الفرعى فإنه يطبع القائمة

الأولى يليها القائمة المخزنة فى READLIST .

● ماذا يحدث بداخل الكومبيوتر ؟

الأشكال التالية من (أ) إلى (ز) تشرح بالتفصيل ما يحدث من عمليات بداخل الكومبيوتر عند تنفذ السطر الثانى من البرنامج :

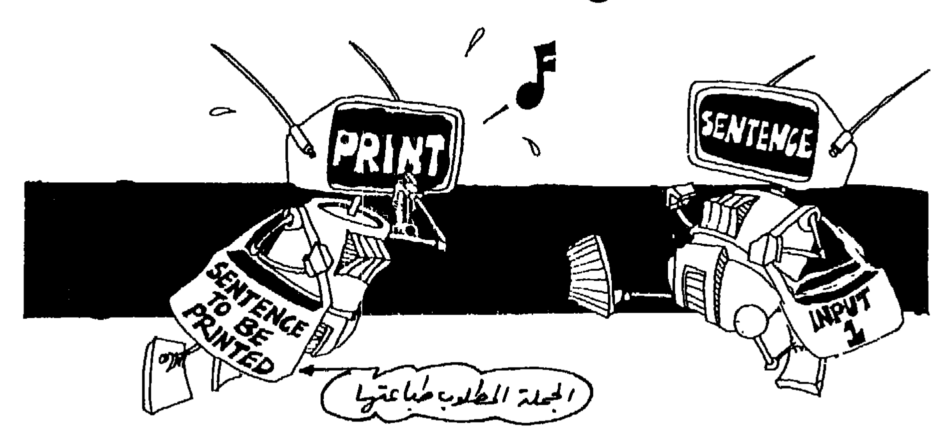
PRINT SENTENCE [YOU JUST MADE ME SAY] READLIST

هذا السطر يتكون من ثلاثة أوامر لوجو وهى :

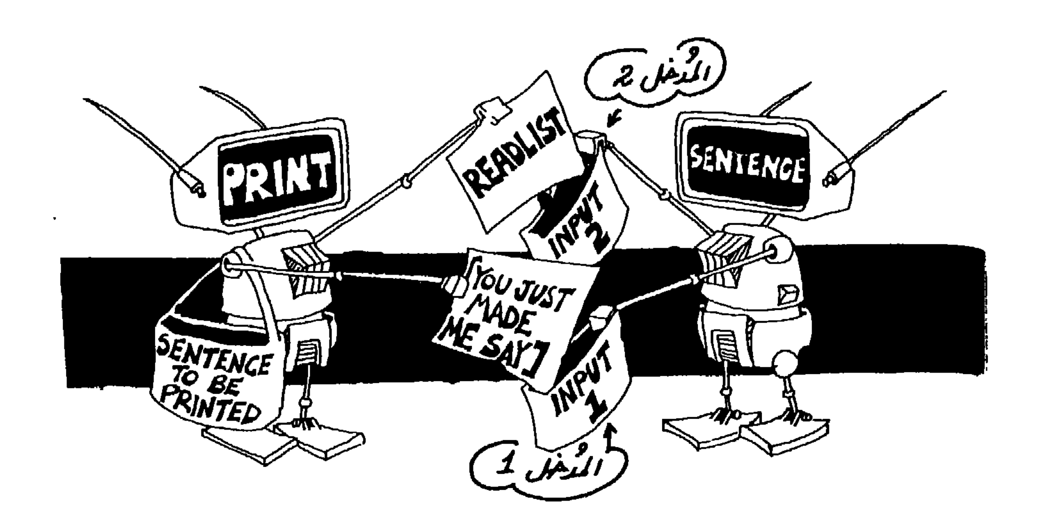
PRINT ، SENTENCE ، READLIST .

وكل أمر من هذه الأوامر يحتاج لبعض المُدخلات (inputs) لكى يؤدى العمل المطلوب منه . فالأمر PRINT مثلاً يحتاج إلى مُدخل يخبره بالشيء المطلوب طباعته على الشاشة .

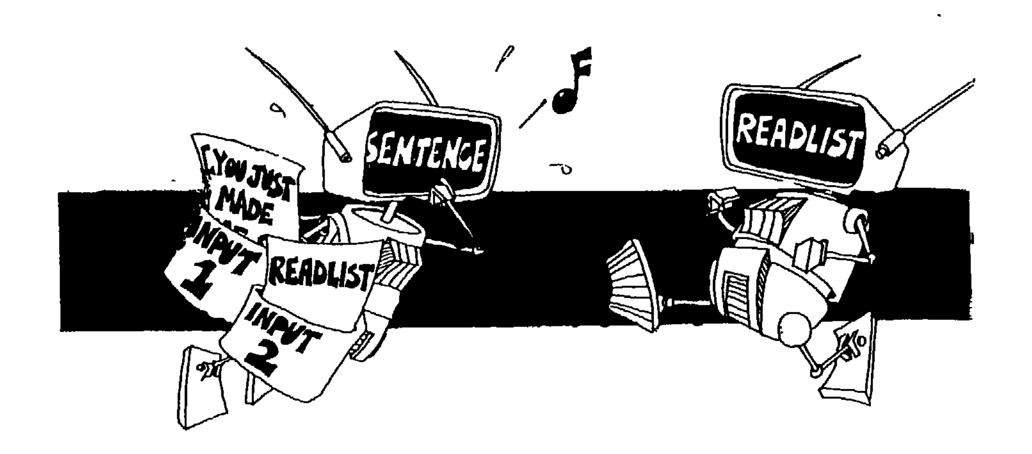
وفى الشكل تم تمثيل كل أمر من هذه الأوامر بروبوط يسعى لجمع المُدخلات اللازمة له حتى يستطيع أن يؤدى عمله .. فلنر معاً ماذا يحدث لتنفيذ السطر الثانى من البرنامج :



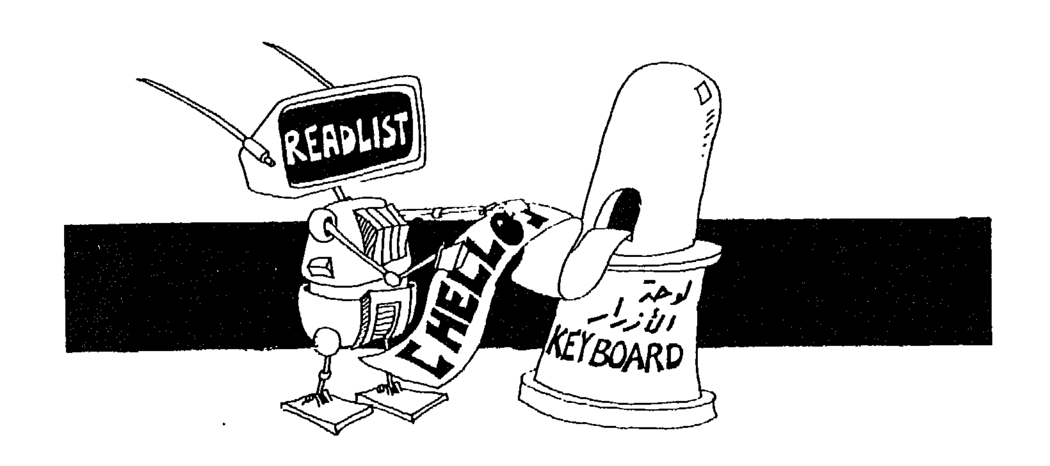
(أ) يحتاج الأمر PRINT لمُدخل ما (input) لكى يعرّف الكومبيوتر ماذا سيطبع . وهو يحصل على هذا المُدخل من الأمر SENTENCE . وهذا هو سر الترحيب الذى يبدو على الروبوط PRINT عندما عثر على المُدخل SENTENCE .



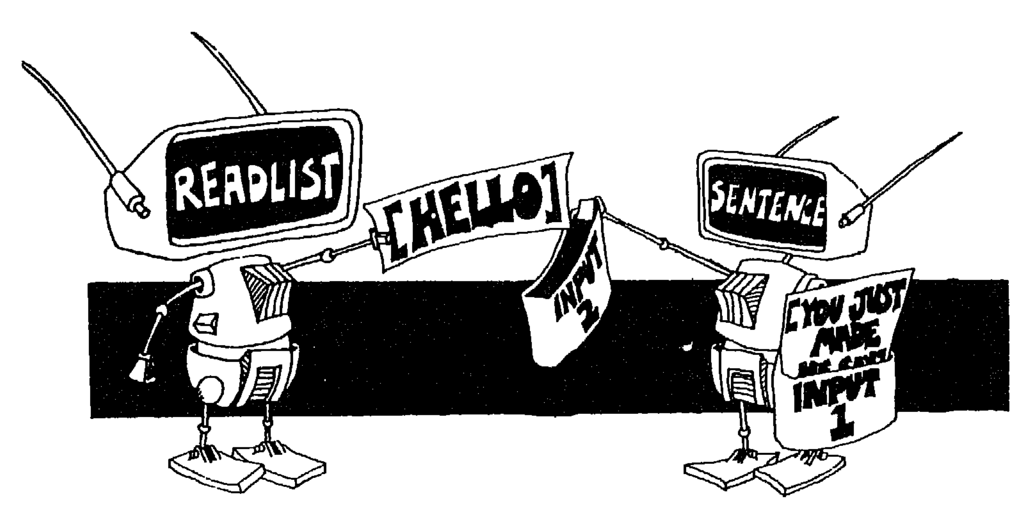
(ب) ولكن الأمر SENTENCE يحتاج هو الآخر لمُدخلين . وفى مثالنا هذا يكون المُدخل الأول هو القائمة [YOU JUST MADE ME SAY] أما المدخل الثانى فهو الأمر READLIST . القائمة جاهزة ولكن الأمر READLIST يجب البحث عنه .



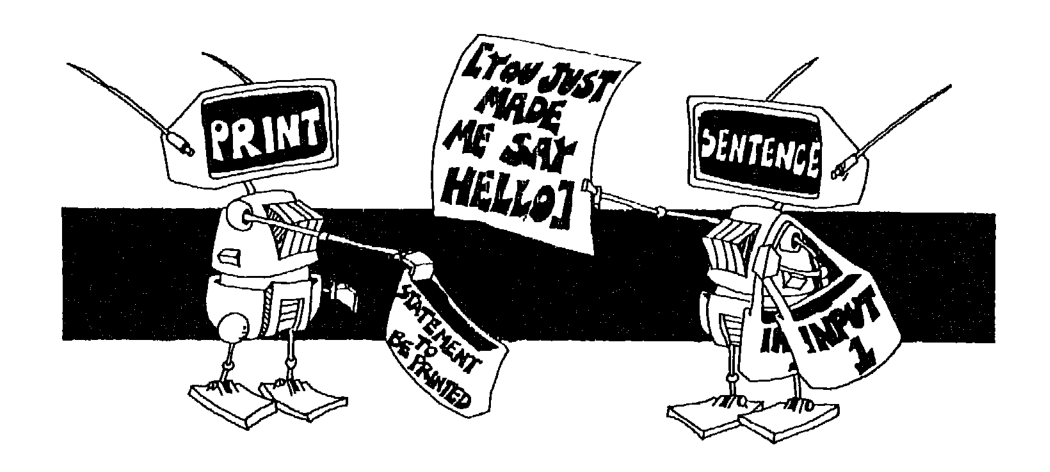
(جـ) ها هو الروبوط SENTENCE يعثر على ضالته المنشودة READLIST .



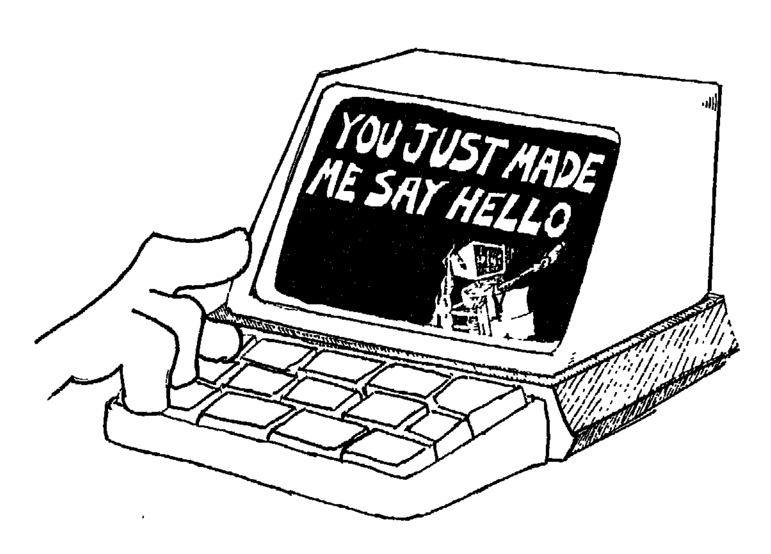
(د) ولكن REALIST يحتاج هو الآخر لمُدخل من لوحة الأزرار حتى يصلح لأداء وظيفته .. وها هو يحصل على المدخل HELLO من لوحة الأزرار .



(ﻫ) الروبوط READLIST يسلّم SENTENCE القائمة [HELLO] ليضعها فى حقيبة المُدخل رقم 2 .



(و) اكتملت الآن القائمة التى يحتوى عليها SENTENCE لذلك فهو يسلمها إلى PRINT ليطبعها على الشاشة .



المرحلة الأخيرة من القصة .. نعرفها نحن جيداً .. بضغطه واحدة على الزر ENTER تطبع العبارة المطلوبة على الشاشة ... هكذا ببساطة .. ربما دون أن نحس بقيمة المجهود الذى بذله كل من READLIST و SENTENCE و PRINT !

● الكومبيوتر يناقش !

● مثال [٢] :

وهنا نضيف إضافة صغيرة تجعل الكومبيوتر ــ بعد أن ينفذ العمل المطلوب منه ــ يعود إليك من جديد قائلاً :

« ولكننى أكره أن أقول هذه الكلمة أعطنى شيئاً آخر لأقوله » .

وكلما أدخلت إليه عبارة عاد من جديد .. يعترض ويريد عبارة أخرى .
وهذا هو البرنامج الفرعى وهو يحمل الاسم BACKTALK :

```
TO BACKTALK
PRINT [PLEASE TYPE SOMETHING FOR ME TO SAY]
PRINT SENTENCE [BUT I HATE TO SAY] READLIST
BACKTALK
END
```

العودة إلى أول البرنامج

القائمة المُدخلة

كيف يعود الكومبيوتر إلى أول البرنامج ليسألك مرّة بعد مرّة أن تدخل له جمله يقولها ؟

إن هذا قد تم بفضل الأمر الذى جاء فى السطر قبل الأخير وهو اسم البرنامج الفرعى نفسه BACKTALK بمعنى تنفيذ البرنامج من البداية .

وهذه الحلقة التكرارية (loop) لا تتوقف إلاّ إذا تدخلت فى البرنامج من الخارج بالاستعانة بالزر Ctrl-Break وذلك للكومبيوتر IBM أو الزر المناسب فى أى كومبيوتر آخر .

● الكومبيوتر موافق دائماً :

● مثال [٢] :

فى هذا المثال فالكومبيوتر يوافق على ما تقدم له فهو يطلب منك أن تكتب له شيئاً تحبه وشيئاً تكرهه وفى الحالتين يبدو موافقاً تماماً على رأيك لكنه أبداً

يعود من جديد ليسألك أن تكتب له شيئاً آخر .

فى هذا البرنامج تدريب جيد لاستخدام الأمر SENTENGE لوصل القوائم معاً بما فى ذلك القائمة التى يحتوى عليها أمر الإدخال READLIST .

```
TO AGREE
PRINT [TELL ME SOMETHING YOU LIKE]
PRINT (SENTENCE [I LIKE] READLIST [TOO])
PRINT [TELL ME SOMETHING YOU HATE]
PRINT (SENTENCE [I HATE] READLIST [EVEN MORE THAN YOU DO!])
AGREE
END
```

وهذا هو تنفيذ البرنامج على الكومبيوتر سنكلير :

```
?AGREE
tell me something you like
 APPLES
I like APPLES too
please type something you hate
 TO DO NOTHING
I hate TO DO NOTHING even more
han you do !
tell me something you like

STOPPED!!! in AGREE
?
```

تشغيل البرنامج الفرعى

الكومبيوتر يسأل

أنت تكتب الإجابة

تعليق الكومبيوتر

السؤال الثانى

أنت تجيب

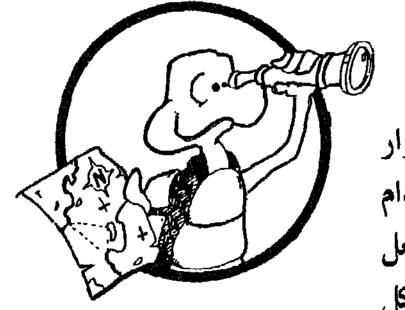
تعليق الكومبيوتر

ويسأل من جديد ...

بإيقاف البرنامج من الخارج

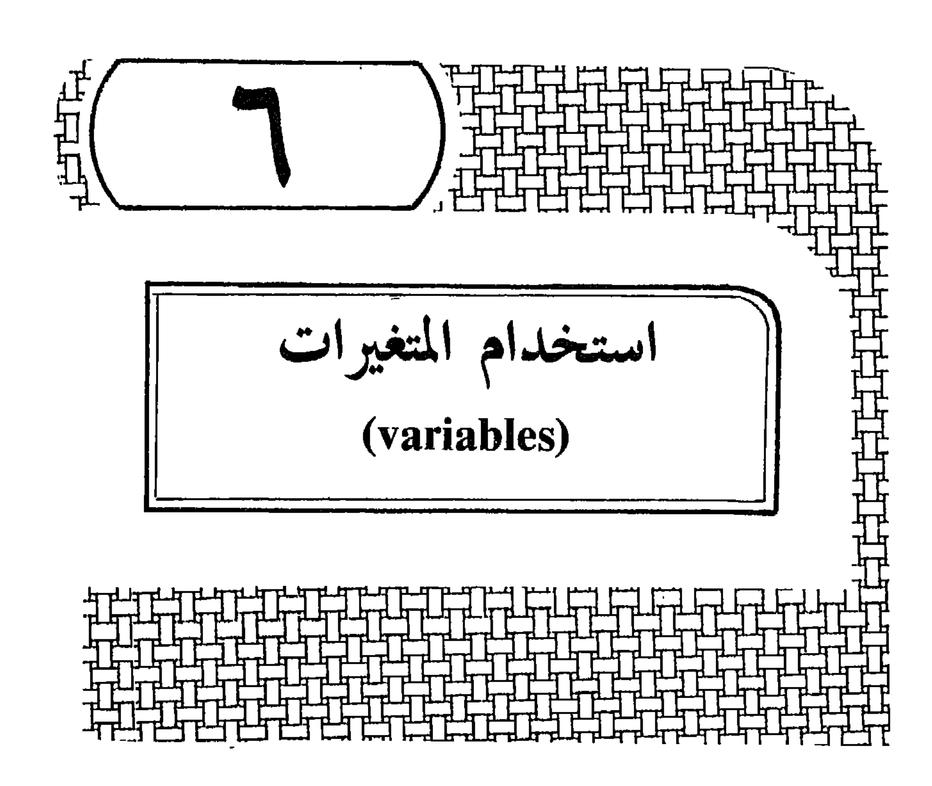
تجربة

الروبوط المعترض دائماً



حاول إنشاء برامج مماثلة للحوار بينك وبين الكومبيوتر باستخدام نفس المنطق . مارأيك ببرنامج يجعل الكومبيوتر دائم الاعتراض على كل شيء ؟

إنه نموذج جيد لروبوط مشاكس .


٦
استخدام المتغيرات
(variables)

● ما معنى المتغيّر ؟

المتغير هو وعاء فى الذاكرة نطلق عليها اسماً ما ونستخدمه لتخزين ما نشاء من البيانات سواء كانت هذه البيانات أرقاماً أو كلمات أو قوائم .

واسم المتغيّر فى لغة لوجو عبارة عن كلمة (WORD) .

ولنر معاً هذا المثال :

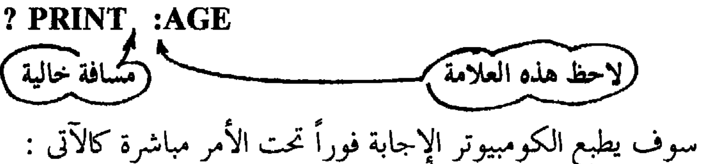
● مثال :

إذا أردنا أن نخزن الرقم 8 فى خانة فى ذاكرة الكومبيوتر فعلينا أولاً أن نمنح هذه الخانة اسماً ، وليكن AGE . ثم نعطى الأمر الآتى :



بهذا الأمر تكون الكلمة AGE محتوية على العدد 8 .

وللتأكد من ذلك فإننا يمكن أن نطلب من الكومبيوتر أن يطبع محتويات خانة الذاكرة المسماة AGE (أو محتويات الكلمة AGE) بالأمر الآتى :



سوف يطبع الكومبيوتر الإجابة فوراً تحت الأمر مباشرة كالآتى :

```
? PRINT :AGE
8
```

الإجابة

● اختر الاسم المناسب للمتغير :

ولكن لماذا اخترنا الاسم AGE اسماً للمتغيّر ؟ فى الواقع أنه يمكن اختيار أى اسم . ولكن إذا كان الرقم 8 يعبر عن عمر أحد الأصدقاء فيكون من المناسب اختيار الاسم AGE للمتغير الذى نحفظ فيه الرقم حتى يكون معبراً عما يحتويه . تماماً كما نحفظ السكر فى علبة مكتوب عليها سكر . والملح فى علبة مكتوب عليها ملح . ولا ضرر إطلاقاً إذا كتبنا على علبة السكر « ملح »

وكتبنا على علبة الملح « سكر » . ولكن ربما إذا لم نتذكر ذلك جيداً أن نضع الملح على الشاى حسب اسم المتغيّر المكتوب على العلبة !

● **الكلمات والقوائم فى متغيرات أيضاً :**

اكتب هذا البرنامج على الشاشة :

```
TO VARS
MAKE "AGE 8
MAKE "NAME [ SALLY OSSAMA ]
MAKE "TEL "5865372
```

فى هذا البرنامج نضع السن (8) فى المتغير AGE ونضع قائمة الاسم [SALLY OSSAMA] فى المتغير NAME أما رقم التليفون (5865372) فنحزنه ككلمة فى المتغير TEL .

فالسن تم تخزينه كعدد .

والاسم تم تخزينه كقائمة لذلك وضعناه بين قوسين مربعين .

والتليفون تم تخزينه ككلمة لذلك استخدمنا قبله العلامة" .

أما المتغيرات جميعاً فهى عبارة عن كلمات لذلك اسبقناها كلها بالعلامة "

(ومن الممكن أيضاً تخزين العدد 8 ككلمة باستخدام العلامة " قبله).بتنفيذ هذا البرنامج بإعطاء الأمر VARS لا يتغير شيء على الشاشة ولكن أشياء حاسمة تأخذ مجراها بداخل الكومبيوتر حيث يتم **تخصيص** الرقم 8 للمتغير AGE ويتم **تخصيص** قائمة الاسم SALLY OSSAMA للمتغير NAME ويتم **تخصيص** الكلمة 5865372 للمتغير TEL .

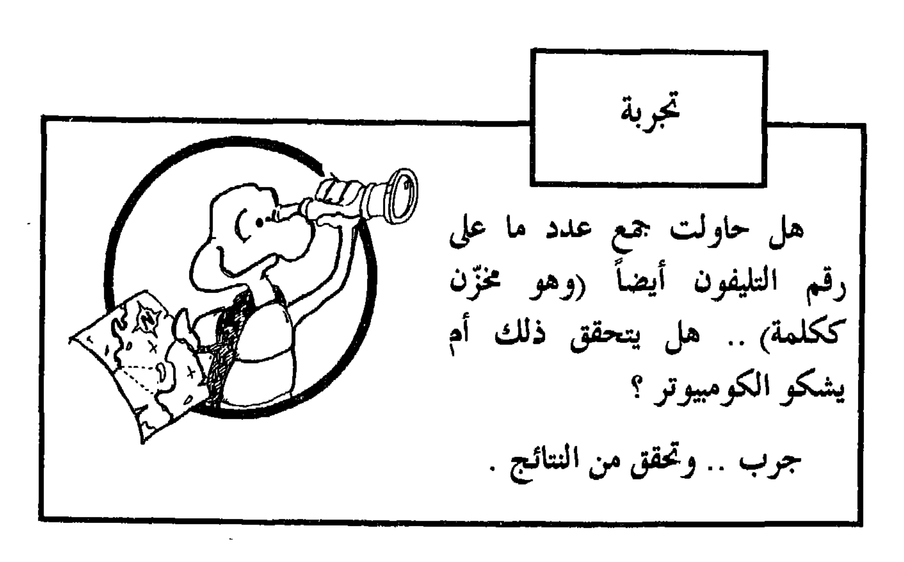
ويمكن التأكد من ذلك بإعطاء الأوامر الآتية بالطريقة المباشرة :

الأمر	**? PR :NAME** ←
الإجابة	**SALLY OSSAMA** ←
الأمر	**? PR :AGE** ←
الإجابة	**8** ←
الأمر	**? PR :TEL** ←
الإجابة	**5865372** ←

والعدد المخزن فى الذاكرة كمتغير يمكن إجراء عليه ما نشاء من العمليات الحسابية . فمثلاً يمكن أن نكتب الأمر الآتى :

	? PR :AGE + 7
الإجابة	**15** ←

فى هذه الحالة يستبدل الكومبيوتر المتغير AGE بالعدد 8 المخزن فيه ويجمعه على العدد 7 .



فى حديثنا السابق استخدمنا كلمة **التخصيص** بدلاً من **التخزين** وهذا هو الاسم الرسمى لعملية تخزين بيانفى متغير .

هل يمكن أن نتحدث الآن عن تخصيص متغير لمتغير ؟ بمعنى تخزين قيمة متغير ما لمتغير آخر باسم مختلف ؟

- **تخصيص متغير لمتغير :**

يمكن إجراء ذلك بأمر كالآتى :

? MAKE "TELPHONE :TEL

فإذا كان الكومبيوتر يعرف قيمة المتغير TEL فإنه يخصصها للمتغير الجديد TELEPHON ويمكن التأكد من ذلك بطباعة قيمة المتغير الجديد TELEPHONE كالآتى :

? PR :TELEPHONE

فلاش

هل هناك ما يحيرك مع استخدام العلامتين (") و (:) ؟

إننا عند التخصيص بالأمر MAKE نستخدم العلامة الأولى .

وعند الطباعة نستخدم الثانية .

ويمكن لعدم الخلط بينهما : فهم العلامة الأولى على أنها « اسم » للمتغير . أما العلامة الثانية فهى تعنى « قيمة » المتغير .

وبذلك يمكن قراءة الأمر الأخير على أنه : « خصص قيمة المتغير TEL لاسم المتغير TELEPHONE .

ومن الممكن استبدال العلامة (:) أمام اسم المتغير بكلمة (THING) وهى نستخدم كالآتى :

? PRINT THING "AGE ← الأمر

8 ← الإجابة

● طباعة العناوين :

قد نرغب فى طبع بعض العناوين أمام البيانات بالصورة الآتية على سبيل المثال :

NAME: SALLY OSSAMA

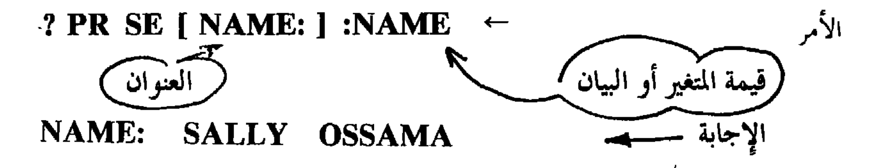
AGE: 8

TEL: 5865372



يمكن إجراء ذلك باستخدام خصائص القوائم التى عرضناها من قبل.فلطباعة العنوان (:NAME) أمام بيان الاسم نكتب الأمر الآتى :

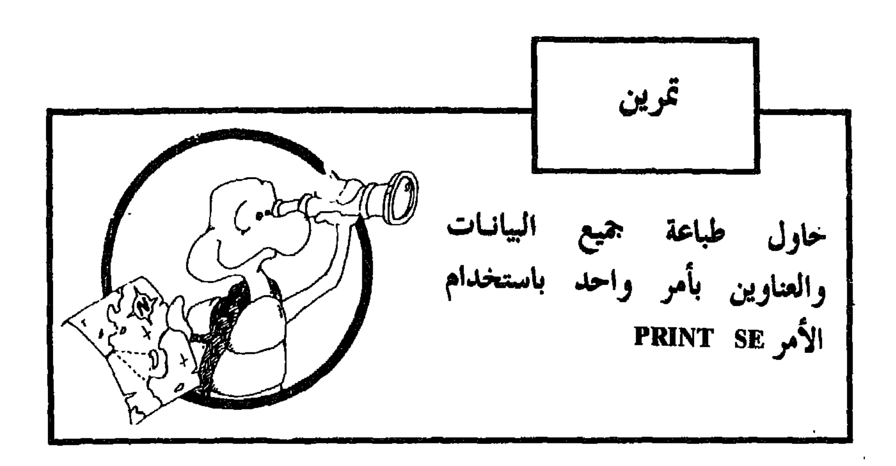
? PR SE [NAME:] :NAME ← الأمر



NAME: SALLY OSSAMA ← الإجابة

وقد تصادف هنا أن يكون العنوان هو (:NAME) ولكن هذا العنوان لا يمت بصلة للمتغير (NAME) وقد كان من الممكن استخدام أى عنوان آخر مثل (NAME OF STUDENT) .

وبالمثل يمكن طباعة بقية البيانات مثل السن والتليفون وأمام كل منها العنوان المناسب .



● الداول **Functions**

هنا نوعية من البرامج تسمى « برامج الدوال » أو « الدوال » . وفى هذه النوعية من البرامج نستخدم المتغيرات ولكن عملية التخصيص للمتغير تتم بطريقة مختلفة تماماً عن الأمر MAKE ولنبدأ بهذا المثال .

● **مثـال :**

اكتب برنامج الدالة الآتى :

```
TO SQUARE :SIZE   (المتغير)
REPEAT 4 [FORWARD :SIZE RIGHT 90]
END
```

هذا هو برنامج المربع (الصندوق) الذى رسمناه من قبل ولكن ظهرت به أشياء جديدة نريد التوقف عندها ، فقد ظهرت كلمة SIZE مرتين بالبرنامج .

فنلاحظ أن الأمر FORWARD لم يعقبه رقم ما كالعادة بل تبعته الكلمة SIZE وهذه الكلمة هى اسم المتغير ، وهو يعبّر هنا عن طول ضلع المربع .

وطول ضلع المربع هنا ليس عدداً ثابتاً بل من الممكن تحديده فى كل مرة يُنفذ فيها البرنامج . وهذا هو الغرض من برنامج الدالة .

ويتميز برنامج الدالة أيضاً بوجود المتغيّر ضمن الإسم :

TO SQUARE :SIZE

وعند تشغيل هذا البرنامج (أو هذه الدالة) فإن الكومبيوتر ينتظر منك إدخال قيمة المتغير SIZE وبمجرد إدخال هذه القيمة يستخدمها فى رسم المربع .

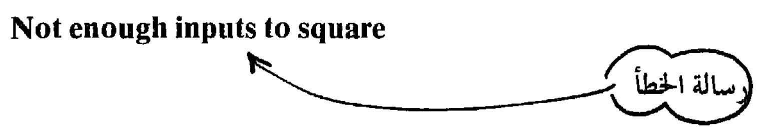
أى أن تشغيل هذا البرنامج يتم بأمر كأحد الأمثلة الآتية :

SQUARE 30
أو SQUARE 40
أو SQUARE 50

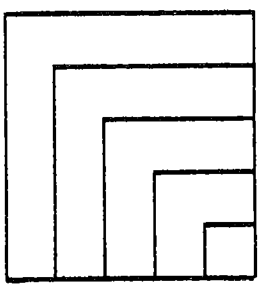
وعند إعطاء الأمر الأول يقوم الكومبيوتر برسم مربع طول ضلعه ٣٠ . أما مع الأمر الثانى فيرسم مربع طول ضلعه ٤٠ وهكذا .. ولكن باستخدام الأمر SQUARE بدون أى رقم بعده يؤدى إلى رسالة خطأ كالمثال الآتى :

? SQUARE

Not enough inputs to square



وبتشغيل هذا البرنامج مرة بعد باستخدام قيمة جديدة للمتغير SIZE يمكن الحصول على شكل كالتالى :



تنفيذ الدالة SOUARE مع تغيير قيمة الضلع كل مرة

والمتغير المستخدم مع الدالة يطلق عليه غالباً « الدليل » (argument) أو « دليل الدالة » .

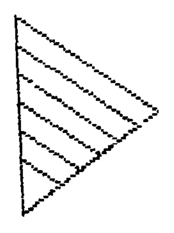
• ارسم مثلثاً متغير الضلع :

هذا البرنامج يرسم مثلثاً متساوى الأضلاع ، وطول ضلعه يمثله المتغيّر SIZE الذى يتم إدخاله كل مرة يتم فيها تنفيذ البرنامج .

طول الضلع

```
TO TRIANGLE :SIZE
REPEAT 3 [FORWARD :SIZE RIGHT 120]
END
```

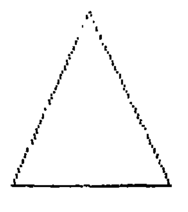
نفذ هذا البرنامج عدة مرات تحصل على مجموعة من المثلثات المتداخلة معاً . مع ملاحظة إخفاء السلحفاه قبل الرسم أو وضع الأمر HT بداخل البرنامج .



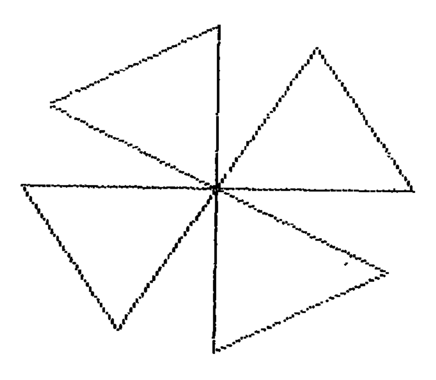
جرب الآن هذا التعديل :

```
TO   TRIANGLE   :IZE
RT   30   FD   :SIZE
REPEAT   2   [ RT   120   FD   :SIZE ]
END
```

فى هذا البرنامج قد أضفنا تعديلاً حتى يُرسم المثلث الموضح بعد ويبدأ الرسم بتغيير الاتجاه إلى اليمين بمقدار 30 درجة .



جرب الآن تنفيذ هذا البرنامج أربع مرات متتالية بنفس طول الضلع SIZE تحصل على الشكل التالى .



تنفيذ البرنامج ٤ مرات بطول ضلع ٧٠
منفذ على كومبيوتر سنكلير

● نداء الدالة لنفسها Recursion

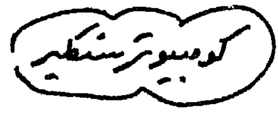
تتميز لغة لوجو بخاصية فريدة وهى إمكان تشغيل البرنامج من داخله .. فالبرنامج السابق للدالة التى ترسم مثلثاً يمكن تشغيله عدة مرات لنحصل على المثلثات المتعددة .

كما يمكن إعطاء الأمر الآتى مرة واحدة لتكرار التنفيذ :

REPEAT 4 [TRIANGLE 70]

فيرسم المثلث ٤ مرات .

أما الشيء الجديد الذى يمكن عمله فهو إضافة سطر أخير فى البرنامج عبارة عن اسم برنامج الدالة نفسه كالآتى :




```
?
TO TRIANGLE :SIZE
RT 30 FD :SIZE
REPEAT 2 [RT 120 FD :SIZE]
TRIANGLE :SIZE
END
```

نداء الدالة

بهذا يتم تشغيل برنامج الدالة بصفة دائمة . فالبرنامج يبدأ برسم المثلث مرة واحدة ثم يأتى للسطر الأخير الذى يرسله إلى أول الدالة فينفذ أوامر البرنامج من البداية وهكذا ..

ولن يتوقف هذا البرنامج إلاّ بالتدخل من الخارج بالضغط على زر التحكم CTRL + break (فى الكومبيوتر IBM) أو على أحد الأزرار المكافئة فى الأجهزة الأخرى .

ولو أن السلحفاه كانت ظاهرة أثناء تنفيذ البرنامج لوجدت أنها تدور فى حلقة مقفلة وهى ترسم نفس الشكل مرّة بعد مرّة .

ولعلنا نلاحظ أنه لا يمكن رسم أكثر من أربعة أشكال فى الشاشة الواحدة أما مازاد عن ذلك فإنه يكون مجرد إعادة للرسم وذلك لأن كل مثلث يحتل ٦٠ درجة علاوة على ٣٠ درجة فاصلة بينه وبين المثلث المجاور وبذلك تحتل المثلثات الأربعة ٤ × ٩٠ = ٣٦٠ درجة .

● **مازلنا نلعب بالمثلثات المتغيّرة :**

● **مثـــال :**

نقدم هنا تطبيقاً آخر للدوال المتكررة التى تستدعى نفسها ولكننا هذه المرّة سوف نحصل على شكل مختلف عند كل تنفيذ للبرنامج .

والبرنامج يتكون من ثلاثة برامج فى الحقيقة :

البرنامج الأول لإعداد الشاشة ويسمى SETUP :

```
TO   SETUP
CS   ←                    تنظيف الشاشة
LT   30   ←               الاستدارة إلى اليسار ٣٠ درجة
HT      ←                 إخفاء السلحفاه
END
```

البرنامج الثانى برنامج فرعى لرسم مثلث ويسمى TRI (اختصار كلمة TRIANGLE) :

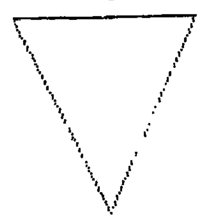
```
TO   TRI
REPEAT 3   [ FD   80   RT   120 ]
END                       لرسم مثلث طول ضلعه ٨٠
```

البرنامج الثالث برنامج دالة يستدعى البرنامج السابق TRI كما يستدعى نفسه ويسمى PAT (اختصار كلمة PATTERN) :

```
TO   PAT   :N
TRI   ←                     استدعاء البرنامج الثانى
RT   360   /   :N   ←       الدوران لليمين
PAT   : N   ←               استدعاء الدالة لنفسها .. للتكرار
END
```

أما البرنامج الأول فهو لمجرد إعداد الشاشة وإدارة السلحفاه بمقدار ٣٠ درجة جهة اليسار وعند تنفيذه بالأمر SETUP لن نشاهد شيئاً على الشاشة لا سيما أن السلحفاه تكون مختفية أيضاً .

عند تشغيل البرنامج الثانى TRI سوف يرسم مثلثاً متساوى الأضلاع قاعدته لأعلى ورأسه لأسفل عند بيت السلحفاه . وطول ضلعه ٨٠ خطوة كما هو موضح بالبرنامج وهذا بعد تنفيذ البرنامج (SETUP) .



أما البرنامج الثالث فهو برنامج للدالة يستدعى البرنامج TRI ويستخدمه ضمن خطواته ولذلك يجب تشغيل هذا البرنامج بعد البرنامج SETUP مباشرة . والدالة PAT تستدعى البرنامج TRI لرسم المثلث السابق أولاً ثم تدير السلحفاه بزاوية مقدارها (360/N) حيث N هو المتغير الذى نكتب قيمته عند تشغيل البرنامج PAT كالمثال الآتى :

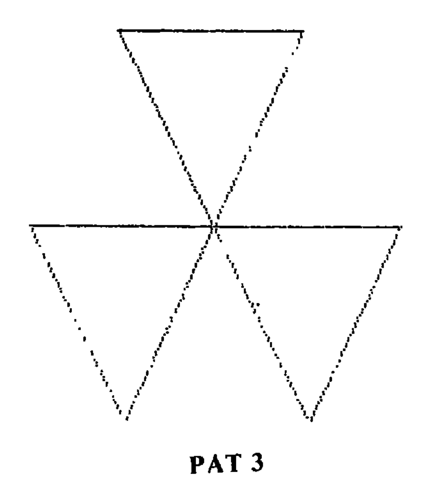
PAT 1

أو **PAT 6**

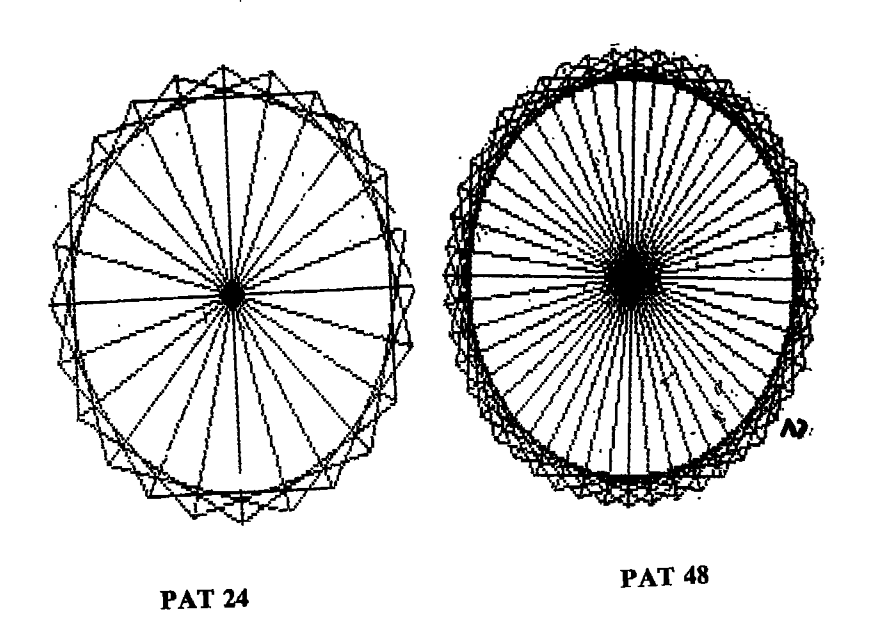
أو **PAT 8**

معنى الدوران بزاوية (360/N) أنه يتم تقسيم الزاوية الدائرية كلها إلى عدد N من الأقسام . فإذا أعطينا الأمر الأول PAT 1 سوف نحصل على مثلث واحد مماثل للمثلث الذى نحصل عليه من البرنامج TRI . وإذا أعطينا الأمر PAT 2 نحصل على مثلثين . والأمر PAT 3 يرسم ثلاثة مثلثات وهكذا ..

جرب الآن تشغيل البرنامج بالأوامر الموضحة وحاول الحصول على الأشكال التالية بعد بتغيير الرقم N .



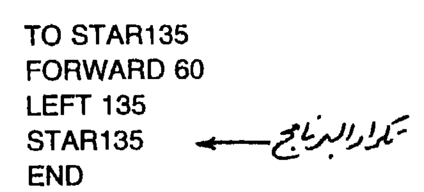
PAT 3

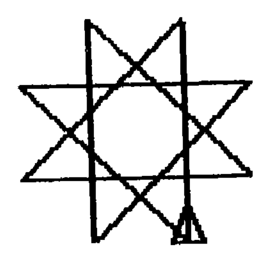


PAT 24 **PAT 48**

● التكرار مع البرامج أيضاً :

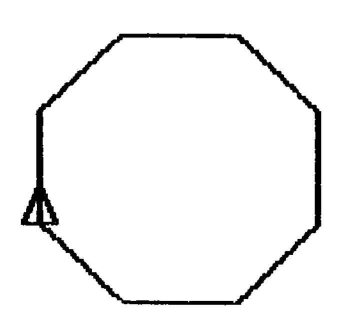
وخاصية التكرار ممكنه مع البرامج العادية أيضاً كما أشرنا من قبل . والمثال الآتى يرسم النجمة ذات الثمانية أضلاع بمجرد رسم أول ضلع لها ثم تحديد زاوية الدوران وإطلاق البرنامج ليكرر نفسه بلا نهاية . والبرنامج إسمه STARI35 باعتبار أن زاوية الدوران لهذه النجمة ١٣٥ درجة .





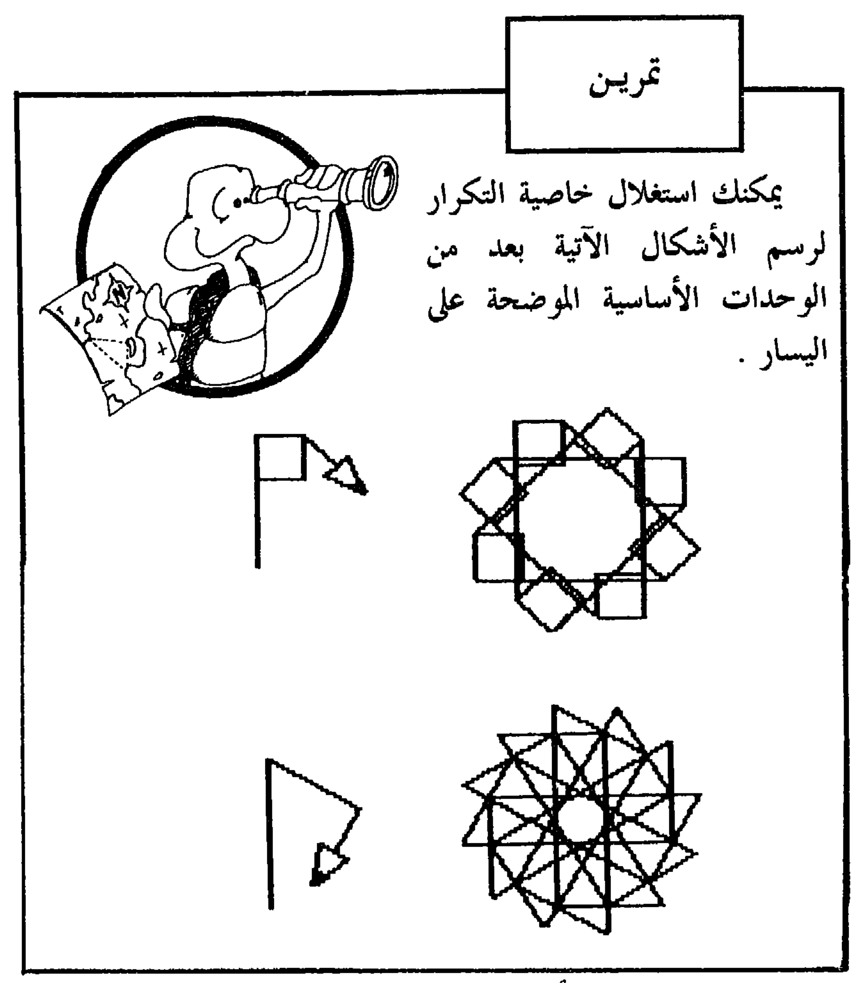
وهذا البرنامج يرسم شكل ثمانى الأضلاع وزاوية الدوران المستخدمة هنا هى (360/8) أى ٤٥ درجة . وهو يرسم أول ضلع ويدير السلحفاه بالزاوية المطلوبة ثم يكرر نفسه .

```
TO STAR45
FORWARD 30
RIGHT 45
STAR45
END
```



تمرين

يمكنك استغلال خاصية التكرار لرسم الأشكال الآتية بعد من الوحدات الأساسية الموضحة على اليسار .

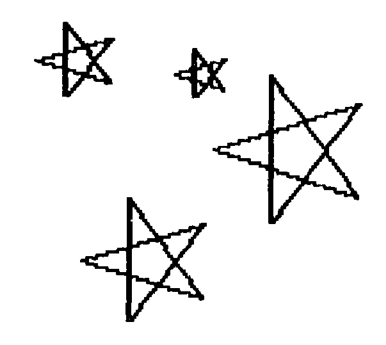


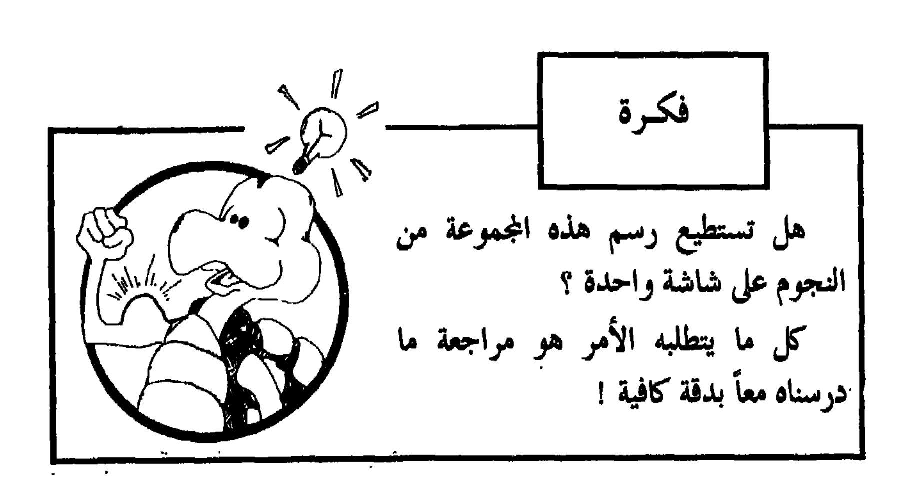
- **ارسم نجمة خماسية بأحجام مختلفة :**

اكتب البرنامج التالى لرسم النجمة الخماسية :

```
TO STAR5 :SIZE
REPEAT 5 [FORWARD :SIZE RIGHT 144]
END
```

ويعمل هذا البرنامج عندما تُدخل الأمر STAR5 متبوعاً برقم ما مثل 50 أو 60 ... إلخ ، يمثل طول ضلع النجمة الخماسية . عندئذ نحصل على أحد الأشكال المرسومة بعد .

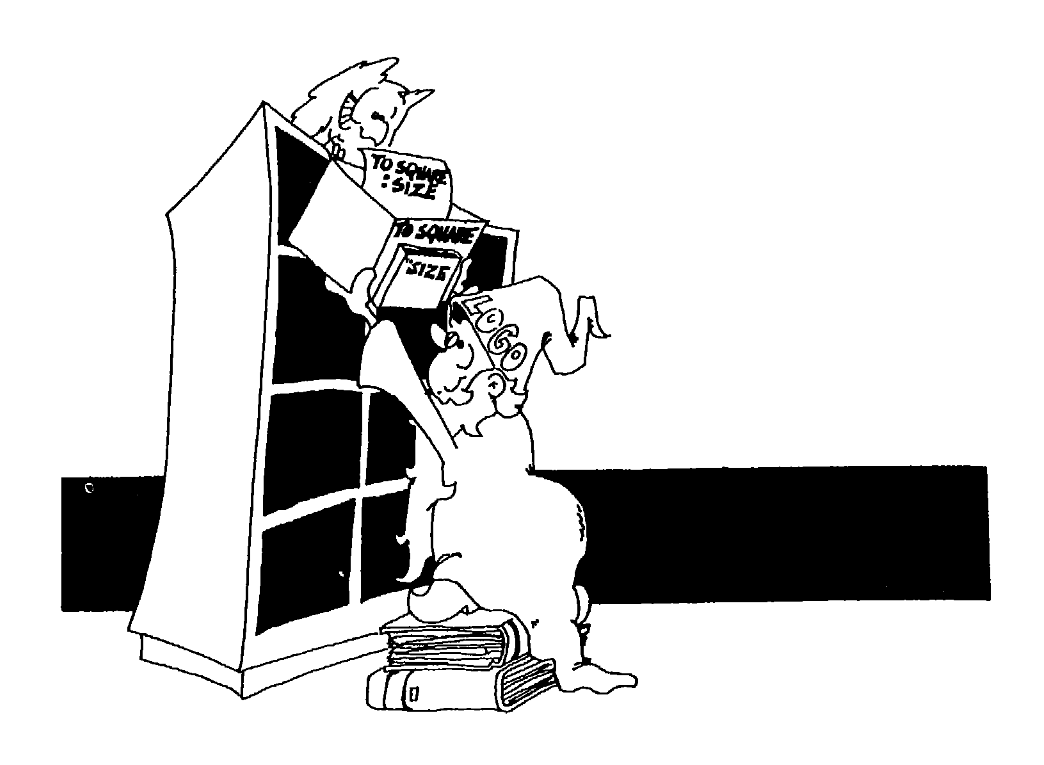




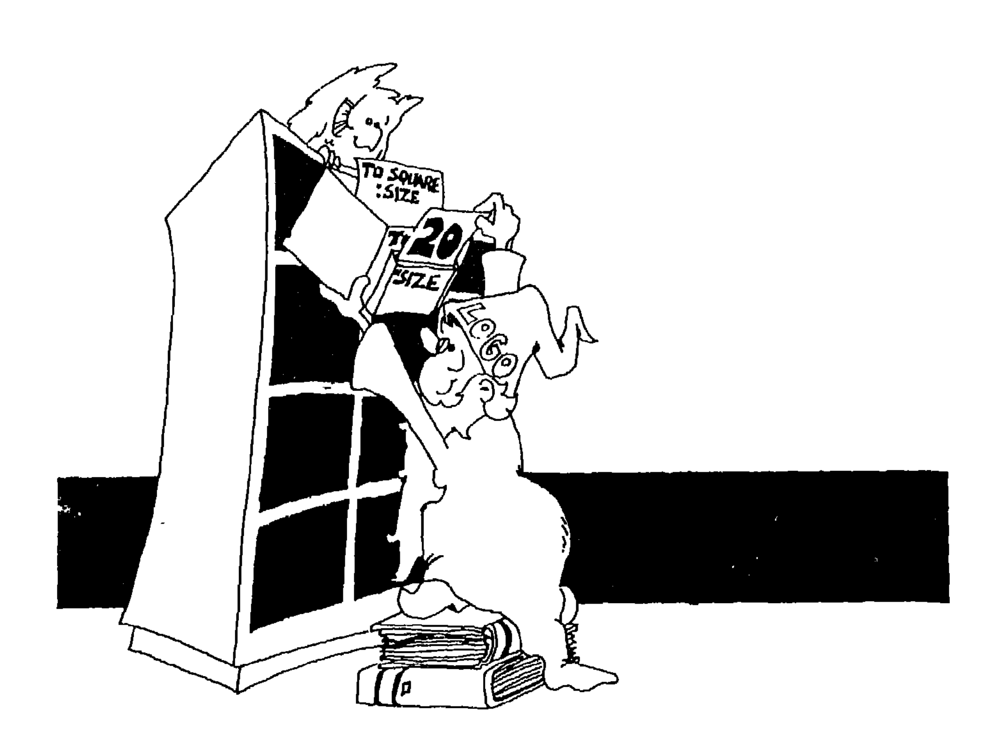
● ماذا يحدث بداخل الكومبيوتر :

تحدثنا عن المتغيرات واستفضنا فى تطبيقاتها سواء باستخدام أمر التخصيص MAKE أو باستخدام برامج الدوال التى ندخل إليها قيمة المتغير (الدليل) .

والأشكال الآتية بعد توضح لنا ما يقوم به الساحر لوجو من أعمال عندما نبدأ فى تعليم الكومبيوتر بالأمر TO SQUARE :SIZE وحتى يتم تنفيذ البرنامج .



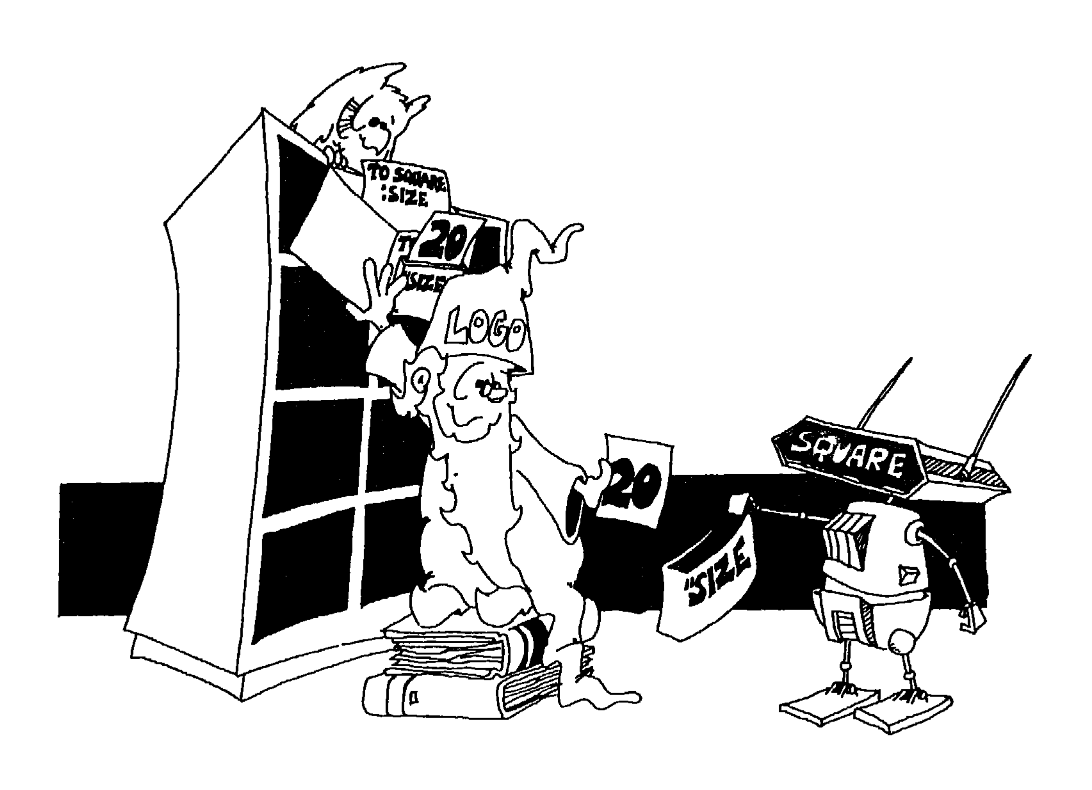
(أ) بعد الانتهاء من كتابة برنامج الدالة (SQUARE : SIZE) فإن الساحر لوجو يحتفظ بخطوات البرنامج فى الذاكرة . ويخصصّ خانة خالية للمتغير "SIZE" ولنطلق عليها خانة الدليل .



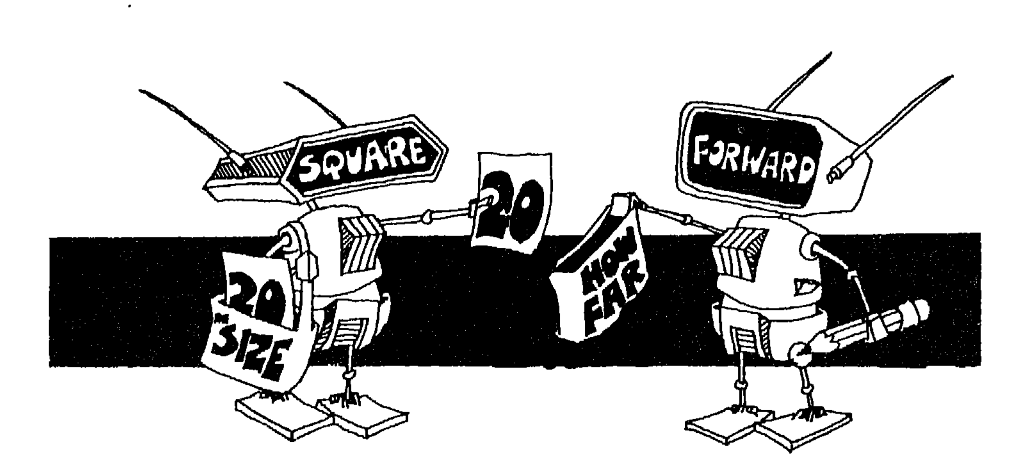
(ب) عندما نعطى أمر تشغيل برنامج الدالة مثل :

SQUARE 20

فإن الساحر لوجو يقوم بوضع الرقم 20 فى خانة الدليل ثم ..



(جـ) يقوم الساحر لوجو باستدعاء الروبوط SQUARE ويسلمه نسخة من قيمة المتغير SIZE“ (أى الرقم 20) حتى يتمكن من تنفيذ العمل المكلّف به .



(د) أثناء تنفيذ برنامج الدالة SQUARE قد يصادف الروبوط SQUARE أمراً مثل SIZE : FORWARD . عندئذ يستدعى الروبوط FORWARD ويسلمه نسخة من قيمة المتغير SIZE ليستخدمها فى أداء عمله .

فلاش

● تذكر الآتى :

"SIZE

يعنى اسم المتغير **SIZE**

أمّا **:SIZE**

فيعنى قيمة المتغير **SIZE**

أمّا **SIZE**

فهو الأمر الذى ندخله للكومبيوتر لكى ينفذ البرنامج SIZE إن وجد .

● **مصطلحات : العلامة (") تسمى بالإنجليزية quotes وبالتالى فإن التعبير "Z يمكن قراءته : quotes Z .**

أما العلامة (:) فهى تسمى dots **وبالتالى فإن تعبيراً مثل N :**

يمكن قراءته : dots N .

● **الفرق بين المتغير العام ومتغيّر الدالة :**

رأينا فى مستهل الموضوع عن المتغيّرات أنه بعد تنفيذ البرنامج المحتوى على الأوامر MAKE ، يمكن طبع قيمة المتغيرات التى تم التخصيص لها ببيانات مختلفة . فمثلاً قد أعطينا الأمر :

```
? PRINT  :NAME
SALLY  OSSAMA
?
```

وكان الجواب حاضراً .

وليس هذا هو الحال مع المتغيرات التى تستخدمها الدالة كدليل لها . بمعنى أنه لا يمكن بعد تنفيذ البرنامج SQUARE : SIZE أن نطبع قيمة المتغير SIZE . فهو يفقد القيمة المخصصة له بمجرد الانتهاء من التنفيذ .

ولنجرب الآتى :

نفذ البرنامج بالأمر :

```
SQUARE  50
```

ثم أعط الأمر التالى :

```
PRINT  :SIZE
```

سوف تكون الإجابة هى رسالة خطأ مثل :

```
SIZE has no value
?
```

فما هو الفرق ؟

إن المتغيّر الذى خصصنا قيمته بالأمر MAKE هو متغير عام يحتفظ به الكومبيوتر فى خانة عامة بحيث أنه يكون موجوداً دائماً فى حيز العمل الذى نحتفظ فيه بكل البرامج والمتغيرات . أما المتغير المستخدم كدليل للدالة مثل SIZE فهو موجود فى خانة تابعة للبرنامج كما رأينا مع الساحر لوجو . ولا يمكن الوصول لمحتويات الخانة SIZE إلاّ من خلال البرنامج أما بعد التنفيذ فإن محتويات الخانة SIZE تُفقد !

وهذه تجربة مثيرة يمكن بها التأكد من ذلك

أعط الأمر التالى :

MAKE "SIZE 50

بهذا الأمر نكون قد كوّننا متغيراً عاماً بالاسم SIZE ووضعنا فيه القيمة 50 . وهو يحمل نفس اسم الدليل SIZE .

وللتأكد من وصول القيمة 50 إلى خانة المتغير العام SIZE أعط الأمر التالى :

الأمر	**? PR :SIZE**
الإجابة	**50**
	?

فى نفس الوقت يمكننا تشغيل برنامج الدالة SQUARE الذى يستخدم الدليل SIZE كالآتى :

SQUARE 20

بهذا الأمر يتم تنفيذ البرنامج مع تخصيص القيمة 20 للمتغير الدليل SIZE .

ماذا تعتقد لو أننا أعطينا الأمر (PR : SIZE) الآن ؟

? PR :SIZE

50

?

إن المتغير العام SIZE ما زال يحتفظ بقيمته 50 ولم يتأثر بالمتغير الدليل SIZE الذى خصصنا له القيمة 20 أثناء تشغيل البرنامج . هذا هو الفرق .

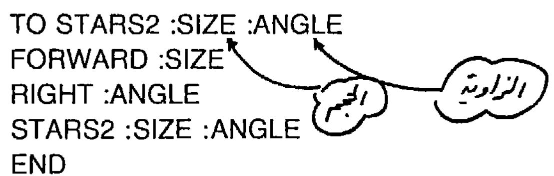
● الدالة بأكثر من دليل :

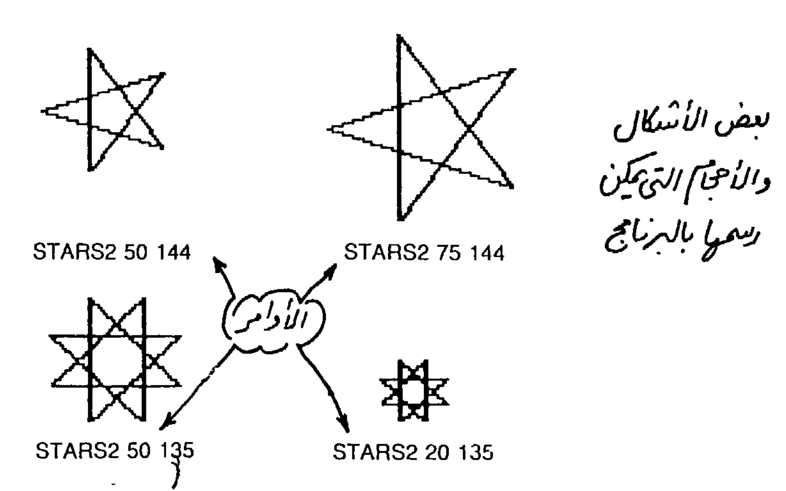
يمكن للدالة أن تأخذ أكثر من دليل وفى هذه الحالة يقوم البرنامج الواحد برسم أكثر من شكل سواء من ناحية الحجم أو التصميم .

فمن الأمثلة التى ناقشناها قبلاً : رسم النجوم المختلفة فهناك النجمة ذات الأضلاع الخمسة التى تحتاج لزاوية دوران ١٤٤ درجة .

وهناك النجمة ذات الأضلاع الثمانية التى تستخدم معها الزاوية ١٣٥ والنجمة ذات الأضلاع التسعة بزاوية الدوران ١٥٠ درجة .

فى الواقع أنه يمكن بناء برنامج واحد لرسم كل النجوم وباستخدام دليلين أحدهما يمثل متغير الزاوية والآخر يمثل طول ضلع النجمة ــ يمكننا رسم أى نجمة نرغبها :





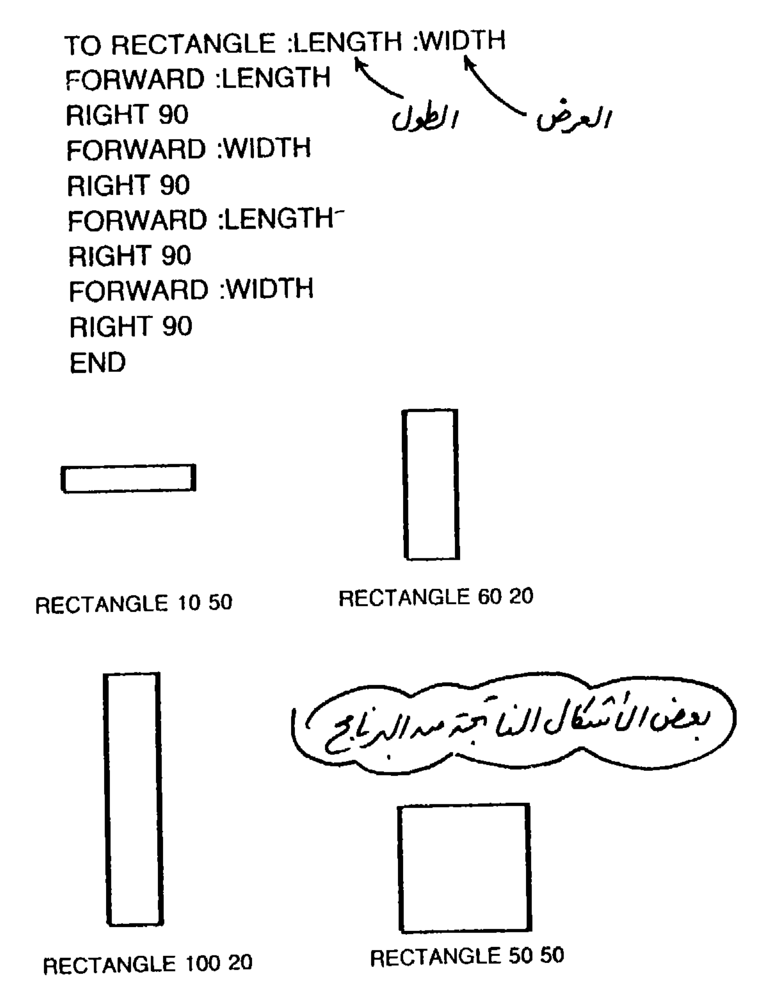
وفى البرنامج الموضح يمثل الزاوية الدليل ANGLE ويمثل الحجم (طول الضلع) الدليل SIZE وعند تشغيل مثل هذا البرنامج يجب إدخال قيمة كل من المتغيرين كالأمثلة الآتية :

```
? STARS2    50   144
? STARS2    60   150
```

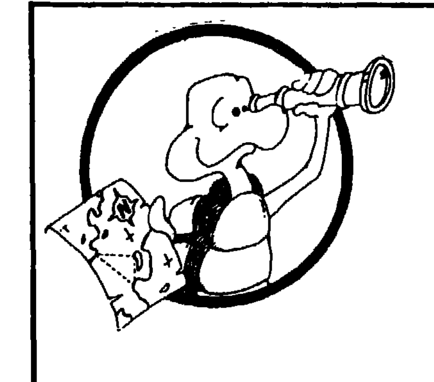
وموضح مع الأشكال التى تلى البرنامج الأوامر التى نتج عنها كل شكل منها .

أدخل ما تشاء من أرقام لهذا البرنامج لتشاهد كل أنواع النجوم الممكنة .

وهذا برنامج يمكن استخدامه لرسم أى شكل رباعى (مربع أو مستطيل) بإدخال الأبعاد المطلوبة للشكل الطول (LENGTH) والعرض (WIDTH) . ويلى البرنامج مجموعة من الأشكال التى يمكن الحصول عليها من البرنامج ومعها الأوامر التى رسمتها .





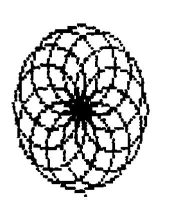


هل يمنك تطويع البرنامج السابق ليرسم أى شكل رباعى (معيّن أو متوازى مستطيلات) علاوة على المربع والمستطيل . حاول وسوف تجد الحل فى الكتاب إذا عجزت .

تطبيقــات

١ أشكال زخرفية بالدوائر :

قدمنا من قبل طريقة رسم هذا الشكل بالبرنامج العادى ، ونقدمه الآن باستخدام برنامج الدالة التى تمنحه مرونة أكثر .



البرنامج الفرعى

```
?
TO CM :S
REPEAT 360 / :S [FD 1 LT :S]
END
```

البرنامج الرئيسى

```
?
TO CX
REPEAT 12 [CM 4 RT 30]
END
```

ولنبدأ بالبرنامج الفرعى CM : S وهو يستقبل المتغير S الذى يمثل زاوية الدوران ويقوم برسم دائرة فى النصف الأيسر من الشاشة .

أما البرنامج الرئيسى فهو يستدعى البرنامج الفرعى ومعه الدليل فى نفس الوقت لذلك فهو يستدعيه بالأمر :

CM 4

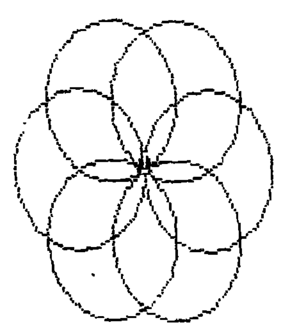
أى أن الزاوية المستخدمة هى ٤ درجات .

وبعد رسم الدائرة تنحرف السلحفاه بمقدار ٣٠ درجة ويتكرر الرسم ١٢ مرة .

ولعلنا نلاحظ العلاقة بين الزاوية ٣٠ درجة وبين عدد مرات التكرار ، حيث أن الرحلة الكاملة ٣٦٠ لا تتسع لأكثر من ١٢ زاوية ٣٠ درجة . ويمكن تكرار الرسم بطريقة أخرى وهى استدعاء البرنامج لنفسه بالأمر CX وذلك بدون تحديد لعدد مرات التكرار وفى هذه الحالة ستظل السلحفاه ترسم وتعيد ما رسمته إلى الأبد !

ولكن المنظر على الشاشة سيظل هو هو,١٢ دائرة . وفى الحالة الأخيرة يوقف البرنامج بالضغط على الزر (Ctrl-Break) أو ما يعادله فى الأجهزة الأخرى (خلاف IBM) .

وإذا كانت الزاوية ٦٠ درجة بدلاً من ٣٠ درجة نحصل على الشكل الآتى الذى قدمناه من قبل وهو يتكرر ٦ مرات فقط .



٢ رسم دائرة ذات نصف قطر معيّن R :

يمكن لأصدقائنا هواة البحث والتقصيّ أن يرجعوا الآن إلى الملحق (١) لدراسة التفصيلات الرياضية لحساب نصف قطر الدائرة .

ومع ذلك فيمكن الآن أن نستخدم البرنامج الآتى لرسم دائرة ذات نصف قطر R .

```
TO  C1  :R  :A
MAKE  "S  2*3.14159*:R / :A
REPEAT  :A [ RT  180/:A  FD  :S  RT  180/:A ]
END
```

حيث R : هو نصف قطر الدائرة المطلوب رسمها .

A : عدد الأضلاع التى تتكون منها الدائرة باعتبار أن الدائرة شكل مضلع . فمثلاً الدائرة التى رسمناها من قبل بالأمر :

```
REPEAT  360 [ FD  1  RT 1 ]
```

تتكون هذه الدائرة من ٣٦٠ ضلعاً طول كل ضلع خطوة واحدة . وأما الدائرة التى ترسم بالأمر :

```
REPEAT  360/10 [ FD  1  RT  10 ]
```

فهى تتكون من ٣٦ ضلعاً طول كل منهم خطوة واحدة أيضاً . وفى البرنامج RCIRCLE أو LCIRCLE للكومبيوتر IBM يستخدم دائماً ٣٦ ضلعاً لرسم أى دائرة . ولكن هذا البرنامج أكثر عمومية وبه حرية اختيار أكثر .

ومع ذلك فيمكنك الإطلاع على البرامج المذكورة فى الملحق رقم (١) واستخدامها كما يمكنك تخزينها على القرص المغنطيسى أو الشريط المغنطيسى واستدعائها عند الحاجة إليها .

وفى الصيغة التى قدمناها الآن نجد أن طوال الضلع (S:) مرتبط بعدد الأضلاع (A:) وبنصف القطر (R:) وبالتالى فإن تغيير عدد الأضلاع (A:) لا يغيّر نصف القطر وإنما يؤثر على درجة الدقة للرسم وعلى سرعة تنفيذه .

وهذا برنامج فرعى آخر يؤدى نفس الغرض ويعتمد على معالجة رياضية مختلفة :

```
TO C2 :R :N
MAKE "S :R * SIN :N
REPEAT 360/:N [ RT :N/2 FD :S RT :N/2 ]
END
```

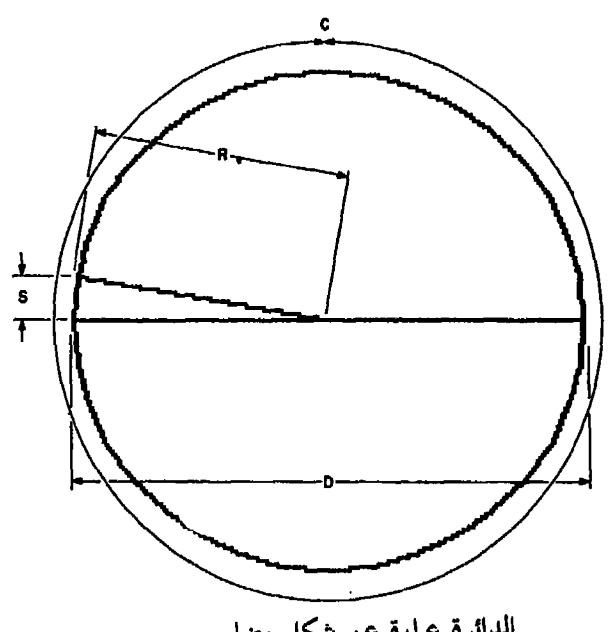
وفى هذا البرنامج تمثل R : قيمة نصف القطر المطلوب .

أما N: فهى الزاوية المواجهة لكل ضلع وهى طريقة مختلفة للتعبير عن عدد الأضلاع فإذا كانت هذه الزاوية ١٠ درجات مثلاً فإن الدائرة تحتوى على ٣٦ ضلعاً .

ويدل S: على طول الضلع .

وفى هذا البرنامج أيضاً لاتؤثر الزاوية (N:) على مساحة الدائرة بل تؤثر على درجة الدقة والسرعة . ولنجرب إدخال القيمة (N:) مساوية للرقم 40 مثلاً فسوف نجد أن الدائرة تحولت إلى شكل ثمانى .

ونلاحظ أن الزاوية يتم تقسيمها إلى نصفين مع الأمر RT وذلك لتحقيق الدقة المطلوبة (راجع الملحق (١)) .



الدائرة عبارة عن شكل مضلع

وللأصدقاء الذين لا يطيقون كثرة التعامل مع التفصيلات فإنه يمكن اختصار أى من البرنامجين السابقين بحيث لا ندخل له سوى نصف القطر فقط R فيصبح كالآتى :

برنامج فرعي لرسم دائرة قطرها R

```
?
TO C3 :R
MAKE "S 3.141592 * :R / 18
REPEAT 36 [RT 5 FD :S RT 5]
END
```

ويعمل هذا البرنامج بكتابة اسمه (C3) يليه نصف القطر المطلوب مثل :

? C3 40

هذا الأمر يرسم دائرة نصف قطرها ٤٠ خطوة .

ولو أنك استخدمت البرنامج C1 فإن الأمر يصبح على سبيل المثال :

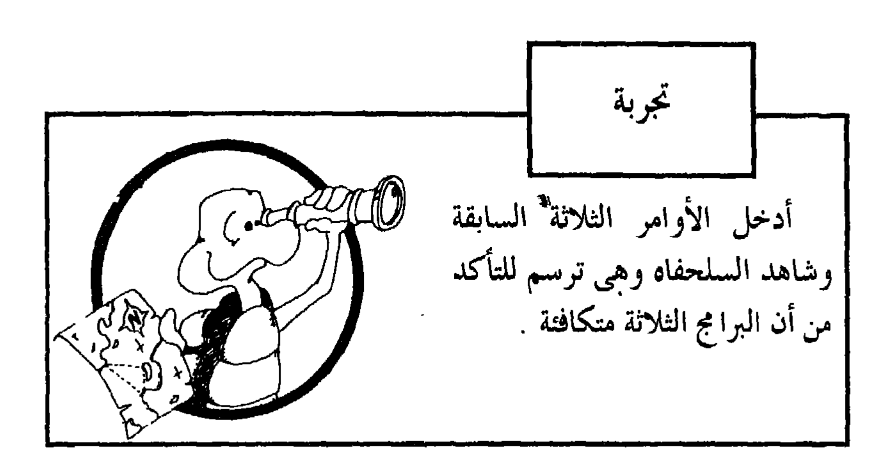
C1 40 36

هذا الأمر يرسم نفس الدائرة وبنفس الدقة .

أما البرنامج C2 فيمكن استخدامه كالآتى :

C2 40 10

وهذا أيضاً يكافىء البرنامجين السابقين :



٣ رسم دائرة فى منتصف الشاشة :

فى البرامج السابقة تحدثنا عن الدائرة على اليمين والدائرة على اليسار والدائرة أعلى أو أسفل الشاشة . ولكننا لم نرسم بعد دائرة فى منتصف الشاشة .

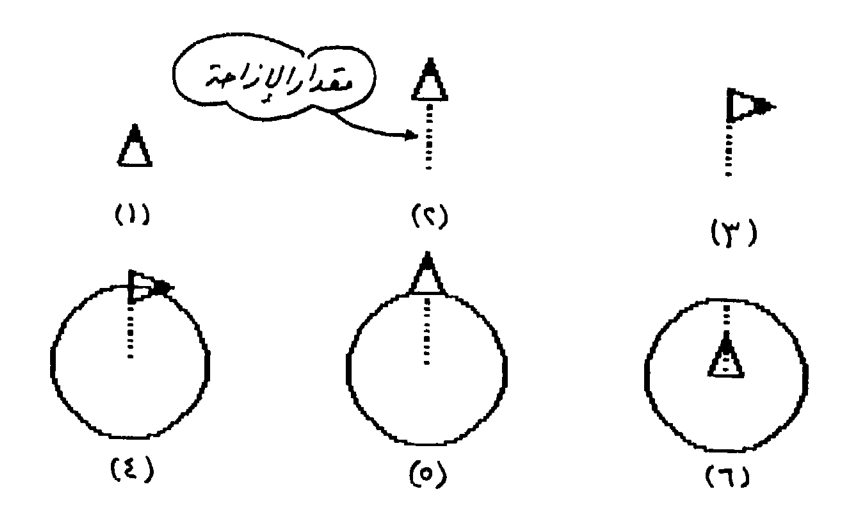
وحيث أن السلحفاه التى ترسم الدائرة تقع على محيطها باستمرار لذلك إذا أردنا أن تكون الدائرة فى المنتصف تماماً فعلينا أن نحرّك السلحفاه قبل البدء فى الرسم إلى المكان المناسب ! ولكن أين هو المكان المناسب ؟ كالعادة أمامنا طريقان :

المحاولة والخطأ **أو** **الحسابات**

ولكن أياً كانت الطريقة المتبعة فهناك اتجاه معين لتحريك السلحفاه . فلو

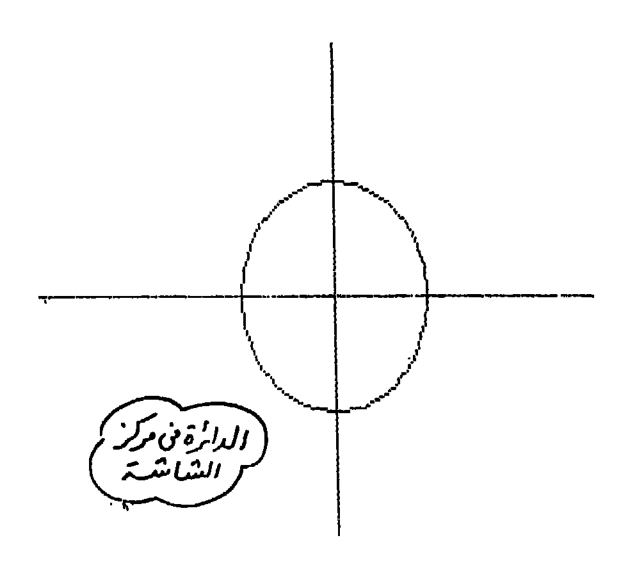
كانت الدائرة على اليمين (أى تستخدم الأمر RT) لزم إزاحة السلحفاه جهة اليسار . ولو كانت الدائرة على اليسار لزم إزاحة السلحفاه جهة اليمين . ولو كانت الدائرة إلى أعلى لزم إزاحتها إلى أسفل وهكذا.. أما مسافة الإزاحة فهى تساوى نصف القطر R .

والشكل التالى يوضح خطوات تحريك السلحفاه إلى أعلى ثم إدارتها إلى اليمين لترسم دائرة (إلى أسفل) وفى النهاية يتم تحريك السلحفاه إلى بيتها الأصلى فنرى الدائرة فى المنتصف تماماً .



رسم دائرة فى منتصف الشاشة

والكومبيوتر IBM يمد بوسيلة جاهزة لهذا الغرض وهى البرنامج الفرعى CCIRCLE الموضح فى الملحق(٢) فى نهاية الكتاب .



ويمكننا استخدام البرنامج CCIRCLE كما هو أو إنشاء برنامج آخر بأنفسنا يؤدى نفس الغرض سواء كان يحمل نفس الاسم أو اسماً مختلفاً .

والبرنامج التالى يقوم برسم الدائرة فى منتصف الشاشة كما هو موضح بالرسم وقد أطلقنا عليه الاسم "C" وهو يأخذ مدخلين A: , R: ومن الممكن بالطبع تحديد القيمة (A:) والاكتفاء بالمتغير R الذى يمثل نصف القطر وبذلك يكون مطابقاً للبرنامج CCIRCLE للكومبيوتر IBM .

وهذا هو البرنامج :

برنامج فرعى لرسم دائرة فى منتصف الشاشة

```
?
TO C :R :A
MAKE "S 2 * 3.14159 * :R / :A
PU FD :R RT 90 PD
REPEAT :A [RT 360 / :A / 2 FD :S
 RT 360 / :A / 2]
LT 90 PU BK :R PD
END
```

ولتبسيط البرنامج أيضاً يمكن التعويض عن عدد الأجزاء (الأضلاع) بالعدد ٣٦ فنحصل على البرنامج فى الصورة التالية :

```
?
TO C36 :R
MAKE "S 3.14159 * :R / 18
PU FD :R RT 90 PD
REPEAT 36 [RT 5 FD :S RT 5]
LT 90 PU BK :R PD
END
```

لرسم دائرة في منتصف الشاشة من ٣٦ ضلعا

ولمقارنة هذا البرنامج بالبرنامج السابق يتم تشغيلهما واحداً وراء الآخر على شاشة واحدة فنحصل على نفس الدائرة .

أظهر السلحفاه أولاً ثم أعط أمر تشغيل البرنامج الأول :

C 40 36

وبعد رسم الدائرة أعط أمر تشغيل البرنامج الثانى :

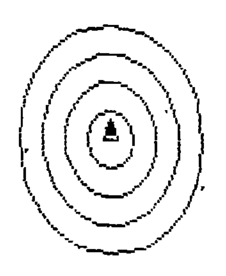
C36 40

سوف نجد أنهما متطابقان .

٤ رسم مجموعة دوائر ذات مركز واحد :

البرنامج التالى لرسم مجموعة من الدوائر المتمركزة باستخدام البرنامج 36 كبرنامج فرعى .

وهذا هو البرنامج مصحوباً بالتنفيذ .



دوائر متمركزة

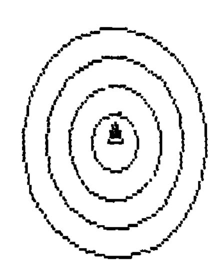
```
?
TO CN
C36 10 C36 20 C36 30 C36 40
END
```

والشكل يوضح مجموعة من الدوائر ذات أنصاف أقطار 10 ، 20 ، 30 ، 40 ، وهى مرادفة للأوامر التى يحتوى عليها البرنامج .

فالأمر 10 C36 يختص برسم الدائرة ذات نصف القطر 10 .

والأمر 20 C36 يختص برسم الدائرة ذات نصف القطر 20 وهكذا ..

.ويمكن رسم نفس المجموعة باستخدام البرنامج C كالآتى :

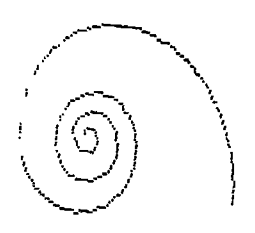


```
TO CON
C 10 36 C 20 40 C 30 50 C 40 60
END
```

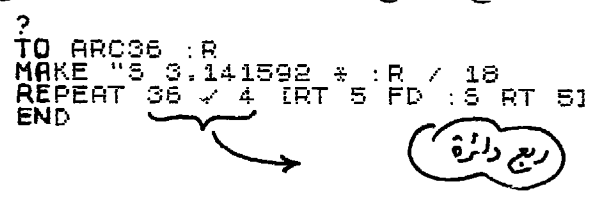
ونلاحظ فى البرنامج استخدام أرقام مختلفة للدليل الثانى وهى لا تغيّر النتيجة من شيء إلاّ فى درجة الدقة وسرعة التنفيذ .

٥ الأشكال الحلزونية :

أما هذا البرنامج فيرسم أشكالاً كالحلزون الموضح بعد :



وهو يتكون من روتين فرعى لرسم قوس نصف قطره متغير (R) وهذا البرنامج هو نفسه البرنامج C36 مع تخفيض عدد مرات التكرار إلى الربع .

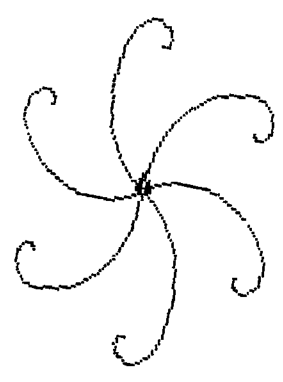


```
?
TO ARC36 :R
MAKE "S 3.141592 * :R / 18
REPEAT 36 / 4 [RT 5 FD :S RT 5]
END
```


ويعمل هذا البرنامج الفر ، بإعطاء الأمر ARC36 يعقبه رقم ما يمثل نصف قطر الدائرة (القوس جزء من دائرة) . وبتكرار تنفيذ هذا البرنامج مرة بعد مرّة مع تصغير قيمة نصف القطر كل مرة نحصل أشكال حلزونية مختلفة .

وهذا البرنامج مكافىء تماماً للبرنامج الفرعى RARC للكومبيوتر IBM (راجع الملحق (٢)) .

ويمكن استخدامه لرسم شكل أخطبوطى ذى ستة أرجل كالموضح بعد .



```
?
TO SPIRAL
ARC36 40 ARC36 10
ARC36 5 ARC36 2
END
```

```
?
TO SPS
MAKE "Z 60
REPEAT 6 [SPIRAL PU HOME PD LT
Z MAKE "Z :Z + 60]
END
```

وهنا يقوم البرنامج الفرعى الأول (SPIRAL) برسم رجل واحدة من أرجل الأخطبوط ثم يتولى البرنامج الرئيسى SPS تكرار العملية .

والملاحظة التى تجدّ علينا هنا هى العدّاد الذى أنشأناه فى البرنامج SPS . فالبرنامج يبدأ بتخصيص 60 للمتغير S الذى يستخدم كزاوية لإدارة الحلزون SPIRAL بعد رسم كل رجل . ونظراً لأن السلحفاه تعود لمنزلها بعد كل حلزون فيلزم تعديل اتجاهها كل مرة بزاوية تزيد عن سابقتها بمقدار ٦٠ درجة لذلك جاءت العبارة :

MAKE "Z :Z+60

بمعنى : « أضف إلى قيمة المتغير (Z) مقدار (60) وسمِّ الناتج (Z) » .

● ضبط الدوائر على الشاشة SCRUNCH, SETSCRUNCH

ذكرنا من قبل أن الدائرة ربما تبدو على الشاشة كما لو كانت قطعاً ناقصاً ellipse بمعنى أن قطرها فى المحور الأفقى يختلف عنه فى المحور الرأسى . وهذا ناتج عن اختلاف طول خطوة السلحفاه فى الاتجاه الأفقى عنه فى الاتجاه الرأسى والتى نطلق عليها نسبة الواجهة (aspect ratio) .

والشاشة القياسية فى لغة لوجو واجهتها تعطى بالنسبة :

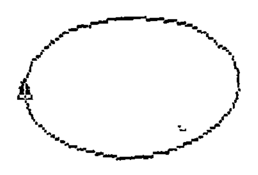
[100 100]

ويمكن الحصول على هذه النسبة بالأمر الآتى :

```
? PRINT SCRUNCH
[ 100 100 ]
```

ويمكن التحكم فى هذه النسبة بالامر SETSCRUNCH .

وبتغيير هذه النسبة يتغير طول خطوة السلحفاه فى أحد الاتجاهين أو كليهما . ففى البرنامج التالى سبق رسم الدائرة ضبط النسبة لتكون [50 100] فنتج عن ذلك رسم القطع الناقص الموضح :



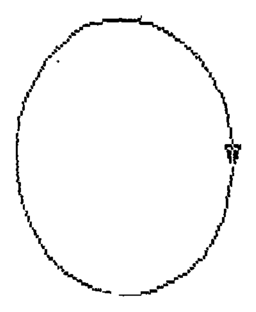
```
?
TO C
SETSCRUNCH [100 50].
REPEAT 36 [fd 50 * sin 10 rt 10]
END
```

فإذا كانت الدائرة أصلاً غير متماثلة على شاشتك فإمكانك استخدام هذا الأمر لتعديل أحد رقمى النسبة بحيث تظهر الدائرة متماثلة . كما يمكن استخدام هذا الأمر أيضاً لضبط المربعات التى تظهر مستطيلة على الشاشة .

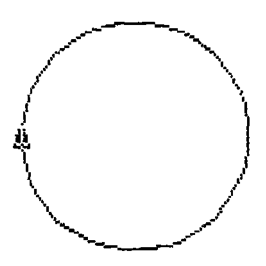
وقد رأينا قبل أن نسبة الواجهة قد تؤثر على شكل الرسم المطبوع أيضاً بحيث تظهر الدائرة سليمة على الشاشة ومنبعجة على الورق .

فى هذه الحالة يتم تعديل النسبة بحيث يُضبط تماثل الرسم على الورق حتى لو بدا غير متماثل على الشاشة .

والمثال الآتى يوضح كيف أن النسبة [100 100] تؤدى إلى طبع قطع ناقص بينما النسبة [80 100] تؤدى إلى طبع دائرة .

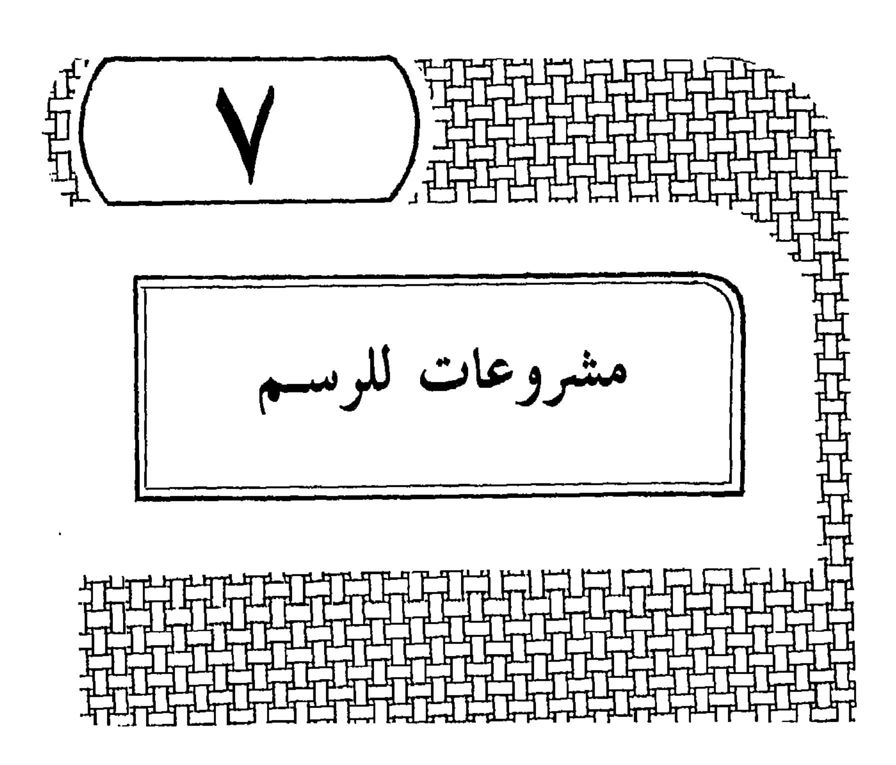


```
?
TO SC
SETSCRUNCH [100 100]
REPEAT 36 [FD 50 * SIN 10 RT 10]
END
```



```
?
TO SC
SETSCRUNCH [100 80]
REPEAT 36 [FD 50 * SIN 10 RT 10]
END
```

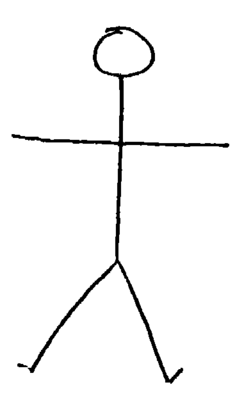
وبتغيير رقمى الواجهة معاً يمكن تكبير وتصغير الرسم بالكامل . جرّب [50 50] , [150 150] وشاهد النتيجة .


٧
مشروعات للرسم

● لنرسم شخصاً :

الخطوة الأولى التى نتخذها عندما نريد أن ننفذ رسماً بالكومبيوتر أن نرسمه رسماً سريعاً باليد « مسوّدة » حتى نعرف الرسم المطلوب بشيء من التفصيل .

وهذا هو رسم اليد :



هذا الرسم مكون من رأس وذراعين ورجلين وجسم ولا بأس من إضافة غطاء للرأس أيضاً (قبعة مثلاً) .

إذن فالبرنامج الرئيسى الذى سنكتبه سيحتوى على البرامج الفرعية التى ترسم كل جزء على حدة . ولنبدأ بكتابة هذا البرنامج الرئيسى PERSON الذى لا يحتوى على أوامر لوجو القياسية وإنما يحتوى على أسماء البرامج الفرعية التى ستقوم بالعمل وهى :

البرنامج BODY لرسم الجسم والبرنامج LEGS للأرجل و ARMS للذراعين و HEAD للرأس و HAT للقبعة .

وهذا النوع من البرامج الرئيسية قد أسميناه قبلاً Superprocedure لأن يرأس جميع البرامج

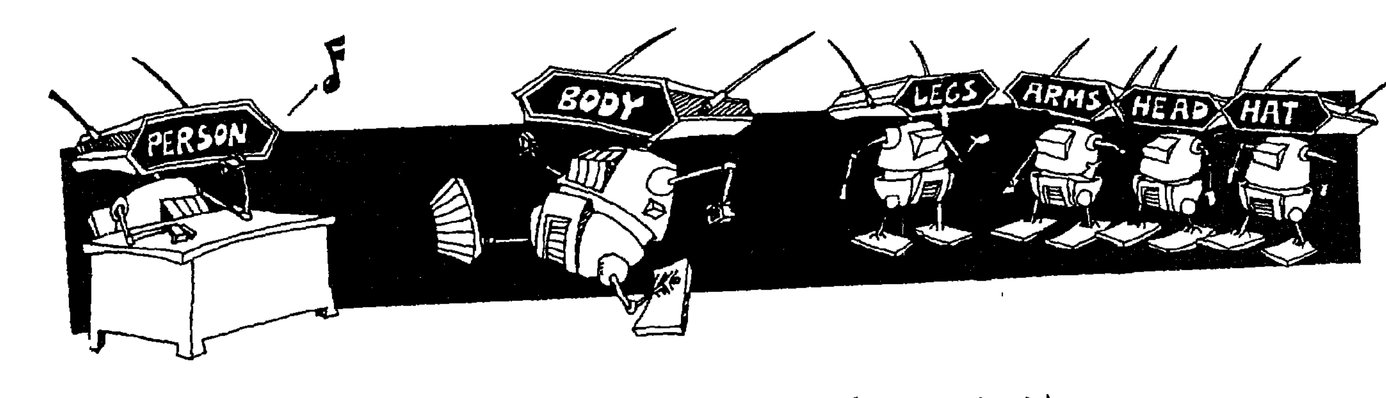
```
TO PERSON
BODY
LEGS
ARMS
HEAD
HAT
END
```

البرنامج الرئيسى
لرسم إنسان

البرنامج الرئيسى

وبذلك **فإن ما يحدث بداخل الكومبيوتر** لا يخرج كثيراً عن التصور الموضح فى الشكل التالى حيث يجلس الروبوط الرئيس PERSON إلى مكتبة مصدراً الأوامر للروبوطات المساعدة الأخرى والتى تحمل أسماء أجزاء الجسم .

ولا يفوتنا أن نلاحظ أن كلاً من الرئيس والمرءوسين عبارة عن روبوطات مبتكرة ! (الشكل السداسى للشاشة) .



ما يحدث بداخل الكومبيوتر عند تنفيذ البرنامج PERSON

ولا بأس من كتابة البرنامج الرئيسى قبل البرامج الفرعية ولكنه مع ذلك برنامج غير قابل للتنفيذ بعد . لأن الكومبيوتر لا يعرف ما هو BODY وما هو ARMS وإلى آخره . ولو أننا حاولنا تنفيذ البرنامج PERSON الآن سوف يشكو الكومبيوتر كالمثال الآتى :

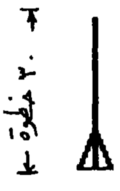
? PERSON

I don't know how to BODY in PERSON

لقد توقف عند أول أمر فى البرنامج وهو BODY .

● فلنبدأ بتعليم الكومبيوتر كيف ينفذ الأمر BODY :

```
TO BODY
FORWARD 30
BACK 30
END
```





إن الجسم المطلوب مجرد خط واحد يعقبه إعادة السلحفاه إلى مكانها بالأمر BACK.

جرب تنفيذ البرنامج الآن . سوف تحصل على الشكل المرسوم ثم يتوقف الكومبيوتر عند الأمر الثانى وهو LEGS مرسلاً الشكوى كالمثال الآتى :

I don't know how to LEGS in PERSON

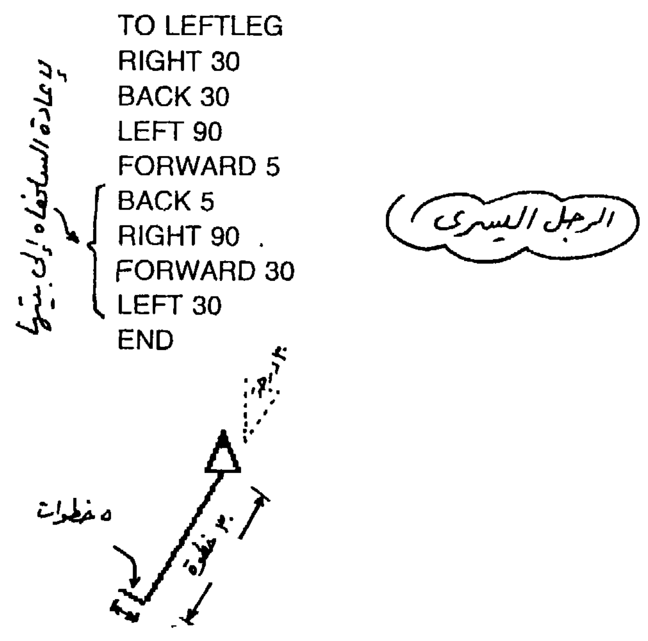
● فلنعلمه كيف ينفذ البرنامج LEGS :

```
TO LEGS
LEFTLEG
RIGHTLEG
END
```



ولكن البرنامج LEGS الممثل للأرجل يحتوى على الرجل اليسرى LEFTLEG والرجل اليمنى RIGHTLEG وكلاهما أمران جديدان أيضاً :

هذا البرنامج يحتاج لبرنامجين فرعيين يرأسهما :

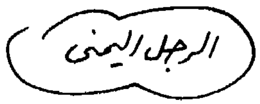



```
TO LEFTLEG
RIGHT 30
BACK 30
LEFT 90
FORWARD 5
BACK 5
RIGHT 90
FORWARD 30
LEFT 30
END
```

والأوامر الأربعة الأخيرة تعيد السلحفاه إلى بيتها ومن الممكن استبدالها بالأوامر التالية : PU HOME PD

● أما البرنامج الخاص بالرجل اليمنى فهو مماثل تماماً للبرنامج السابق مع تبديل كل انحراف جهة اليمين (RT) بنظيره جهة اليسار (LT) والعكس بالعكس .

```
TO RIGHTLEG
LEFT 30
BACK 30
RIGHT 90
FORWARD 5
BACK 5
LEFT 90
FORWARD 30
RIGHT 30
END
```

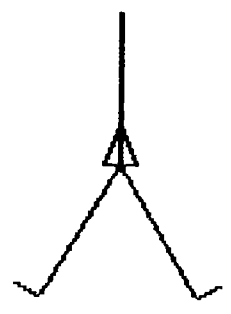


عند تنفيذ هذا البرنامج نحصل على الرسم التالى :

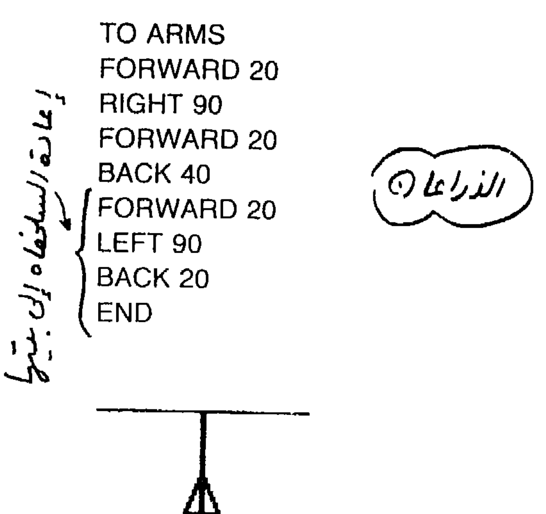


هل نجرب البرنامج الرئيسى PERSON ؟

سوف نحصل على الرسم التالى قبل أن يشكو الكومبيوتر من عدم استطاعته تنفيذ الأمر ARMS .

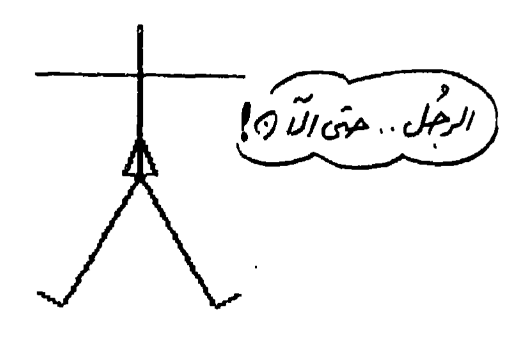


● والآن فلنعلم الكومبيوتر كيف يرسم الذراعين بالبرنامج ARMS :

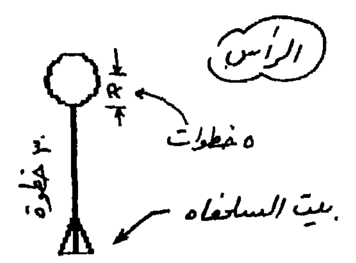


```
TO ARMS
FORWARD 20
RIGHT 90
FORWARD 20
BACK 40
FORWARD 20
LEFT 90
BACK 20
END
```

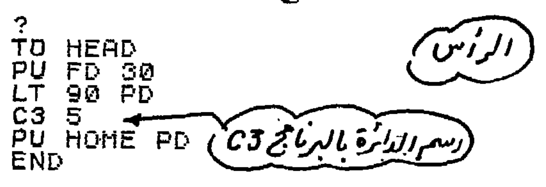
حتى الآن قد حصلنا على هذه الأجزاء الموضحة من جسم الرجل :



● لنكتب الآن برنامج الرأس : HEAD



يمكن هنا أن نستعين بالبرنامج الفرعى (C3) الذى استخدمناه من قبل لرسم الدائرة أو أى برنامج بسيط لرسم الدائرة . أما مستخدمو الكومبيوتر IBM فيمكنهم استخدام البرامج الفرعية الجاهزة الخاصة بالدوائر والأقواس على القرص (LWIL) وسوف نستخدم القيمة 5 كنصف قطر للدائرة التى تمثل الرأس باعتبارها قيمة مناسبة . وهذا هو برنامج الرأس HEAD .



```
?
TO HEAD
PU FD 30
LT 90 PD
C3 5
PU HOME PD
END
```


فى هذا البرنامج يتم رفع القلم تمهيدا للحركة دون رسم . تتحرك السلحفاه إلى أعلى الجسم (FD 30) ثم تدار إلى اليسار ٩٠ درجة مع خفض القلم .

عندئذ يبدأ البرنامج الفرعى (C3) فى رسم دائرة نصف قطرها 5 وبعد الانتهاء يُرفع القلم مرة أخرى وتعود السلحفاه إلى بيتها ثم يُخفض القلم تمهيداً للرسم التالى .

والبرنامج الفرعى (C3) كالآتى :

برنامج الدائرة C3

```
?
TO C3 :R
MAKE "S 3.141592 * :R / 18
REPEAT 36 [RT 5 FD :S RT 5]
END
```

بعد رسم الرأس تكون السلحفاه فى بيتها ورأسها متجهٌ إلى أعلى .

بهذا يكون شكل الرجل قد اكتمل ولا بأس من أن نمنحه قبعة يرتديها كنوع من التدريب على الرسم .

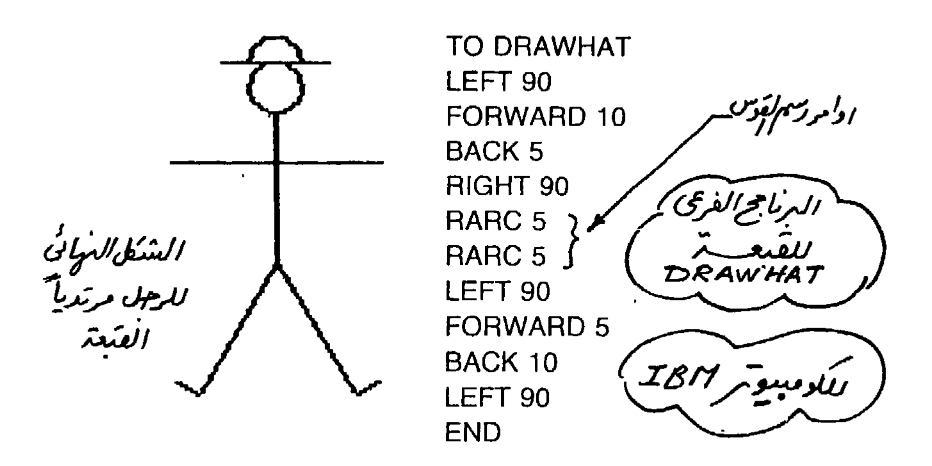
وهذا هو البرنامج HAT لرسم القبعة :

برنامج القبعة

```
TO HAT
PENUP
FORWARD 40
PENDOWN
DRAWHAT   <-- برنامج فرعى
PENUP
BACK 40
END
```

ويشمل البرنامج التحضيرات الأساسية لرسم القبعة وهى الحركة إلى أعلى الرأس (FD 40) والعودة بعد الرسم إلى مركز الشاشة (BK 40) ، أما القبعة نفسها فيتم رسمها بالبرنامج الفرعى DRAWHAT .

والبرنامج التالى هو البرنامج الفرعى لرسم القبعة **وهو خاص بالكومبيوتر IBM** حيث يستخدم البراج الفرعية الخاصة RARC . والشكل النهائى للرجل مرتدياً القبعة موضح مع البرنامج .



فماذا بالنسبة لباقى أجهزة الكومبيوتر ؟ يمكننا ببساطة أن ننشىء برنامجاً فرعيّاً لرسم القوس الذى تتكون منه القبعة ولا مانع من أن نطلق عليه نفس الاسم RARC **وفى هذه الحالة يمكن استخدام برنامج IBM مع أى كومبيوتر آخر .**

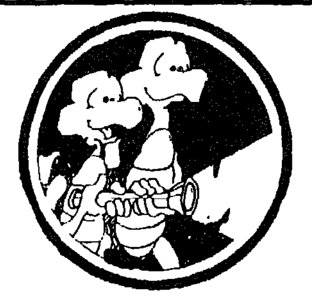
هل نتذكر البرنامج ARC36 الذى قدمناه من قبل لرسم القوس ؟

سوف نستخدمه الآن ونمنحه إسماً جديداً هو RARC حتى يكون مطابقاً لبراج IBM . وهذا هو البرنامج .

```
?
TO RARC :R
MAKE "S 3.141592 * :R / 18
REPEAT 36 / 4 [RT 5 FD :S RT 5]
END
```



فــلاش

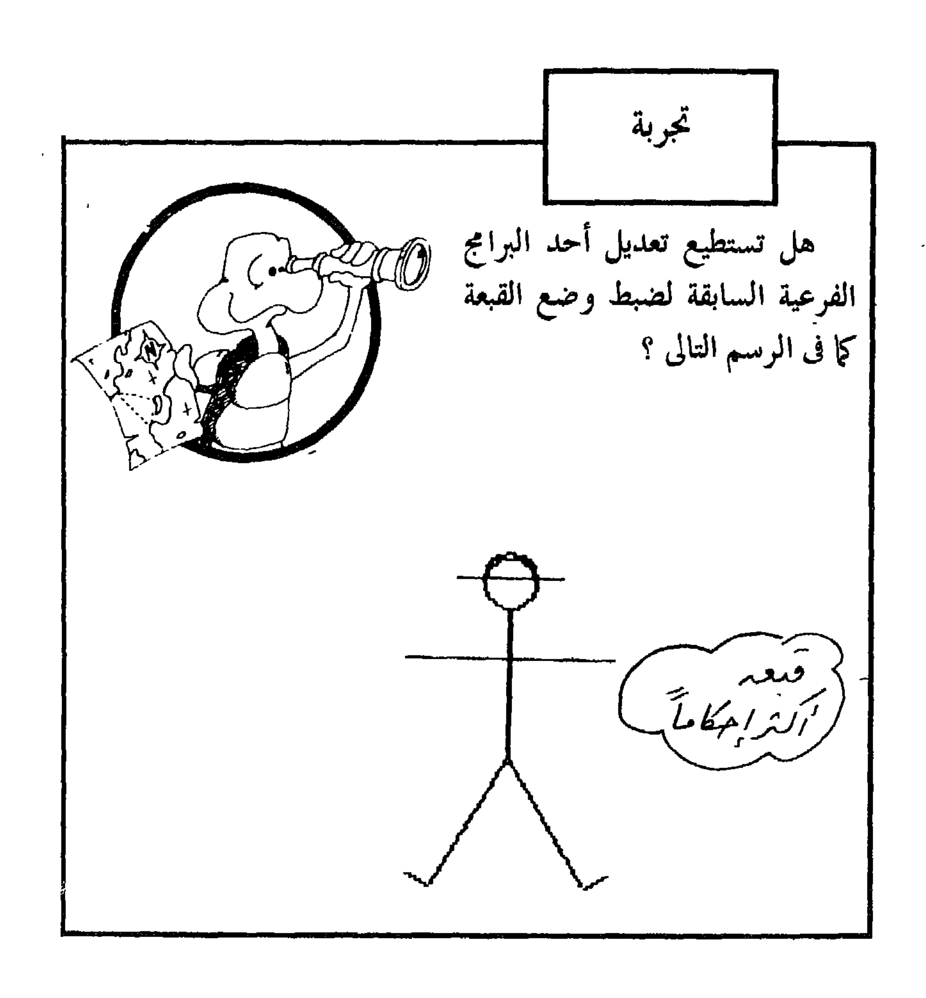


البرنامج السابق يبدأ بالأمر MAKE الذى يحدد قيمة المتغير S .

ماذا تعتقد لو حذفنا هذا السطر من البرنامج ؟

فى الحقيقة أنه لن يتغير شيء وسوف يُرسم الشخص دون أى شكوى من لوجو .

لماذا ؟ ... لقد أشرنا من قبل أن هناك متغيرات عامة يتم اختزانها فى حيز العمل (workspace) عموماً ومتغيرات خاصة بالدوال يقتصر استخدامها على برنامج الدالة فقط . والمتغيّر S من النوع العام ويكفى تعريفه مرّة واحدة فى أى برنامج من البرامج .

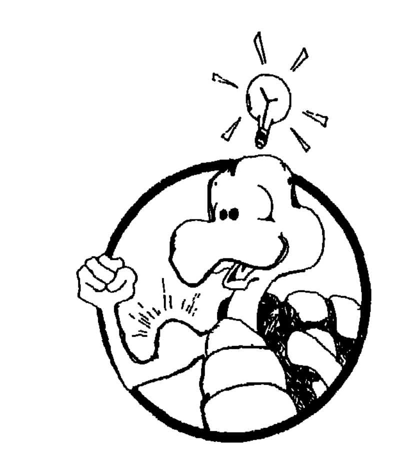


تجربة

هل تستطيع تعديل أحد البرامج الفرعية السابقة لضبط وضع القبعة كما فى الرسم التالى ؟

فكـرة

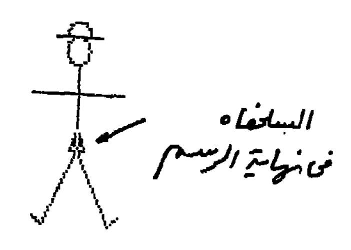
إذا حاولت رسم الشخص مرتين متتابعتين فسوف تلاحظ أنه فى المرة التالية سيختفى جزء من جسم الرجل . علل لذلك . وإذا احتجت لمعاونة يمكنك مقارنة البرنامج التالى بنظيره من البرامج التى عرضناها من قبل .



```
?
TO HAT
PU FD 40
PD DRAWHAT
PU BK 40 PD
END
```

وبمناسبة الحديث عن حيز العمل وما يحتويه من برامج ، نقدم هنا طبعة كاملة لحيز العمل بما يحتويه من برامج ومتغيرات أثناء رسم الرجل .

والآتى بعد لائحة بأسماء البرامج المستخدمة مع الشكل النهائى للرجل .




```
?
TO RARC :R
TO C3 :R
TO DRAWHAT
TO LEFTLEG
TO RIGHTLEG
TO HAT
TO HEAD
TO ARMS
TO LEGS
TO BODY
TO PERSON
```

وفيما يلى نص البرامج التى استخدمت علاوة على المتغيرات التى احتوى عليها حيز العمل بالكامل .

```
?
TO RARC :R
MAKE "S 3.141592 * :R / 18
REPEAT 36 / 4 [RT 5 FD :S RT 5]
END

TO C3 :R
MAKE "S 3.141592 * :R / 18
REPEAT 36 [RT 5 FD :S RT 5]
END

TO DRAWHAT
LT 90 FD 10
BK 5 RT 90
RARC 5 RARC 5
LT 90 FD 5 BK 10 LT 90
END

TO LEFTLEG
RT 30 BK 30 LT 90
FD 5 BK 5 RT 90
FD 30 LT 30
END

TO RIGHTLEG
LT 30 BK 30 RT 90 FD 5
PU HOME PD
END

TO HAT
PU FD 40
PD DRAWHAT
PU BK 40 PD
END

TO HEAD
PU FD 30
LT 90 PD
C3 5
PU HOME PD
END

TO ARMS
FD 20 RT 90 FD 20
BK 40
FD 20 LT 90 BK 20
END

TO LEGS
LEFTLEG
RIGHTLEG
END

TO BODY
FD 30
BK 30
END

TO PERSON
BODY
LEGS
ARMS
HEAD
HAT
END

MAKE "S "0.8726644
```

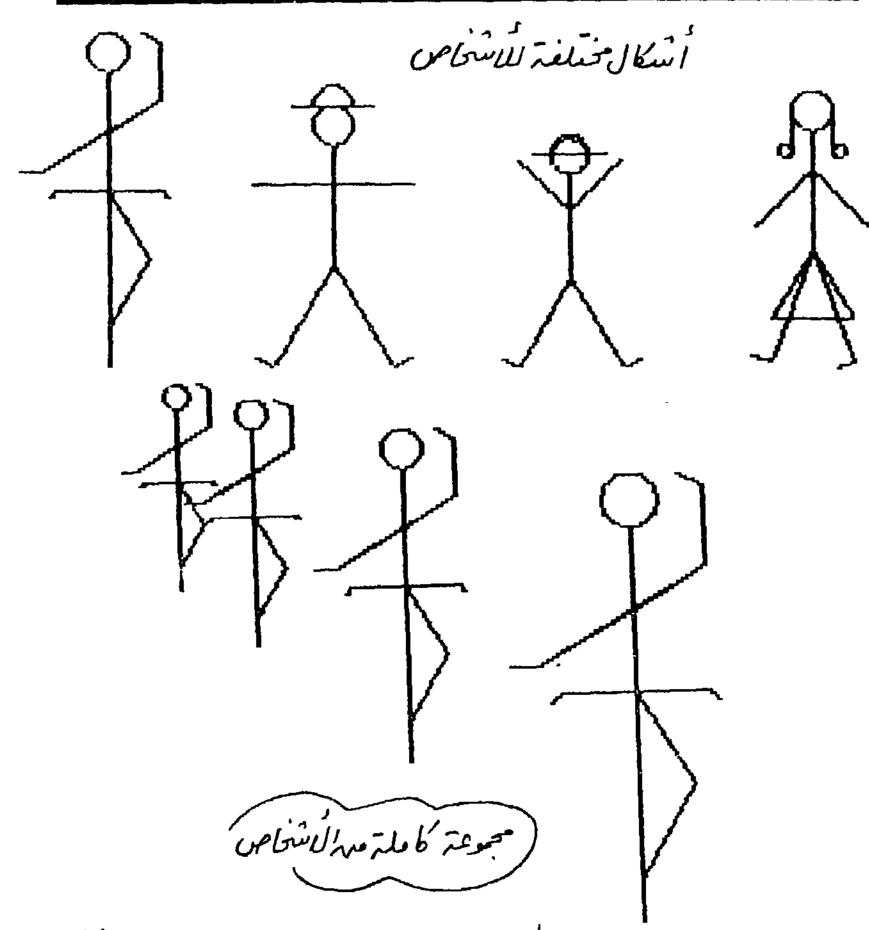
البرامج والمتغيرات في حيز العمل

المتغير العام S

تمرينات

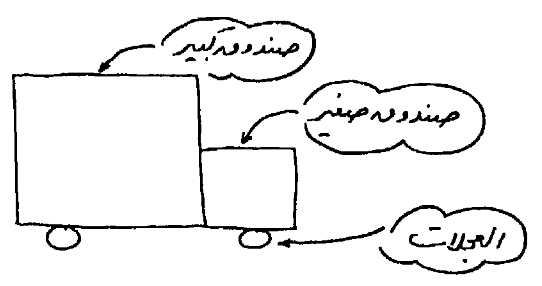


باستخدام البرنامج السابق حاول إجراء التعديلات المناسبة للحصول على الأشكال الآتية :

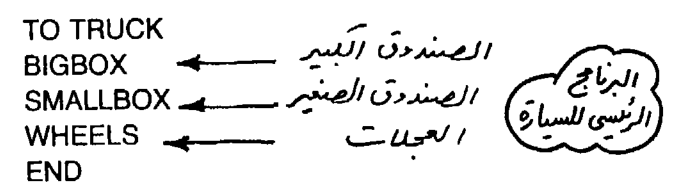


● قافلة من السيارات :

لنبدأ بمسوّدة للرسم باليد نستوضح فيها معالم السيارة المطلوبة :

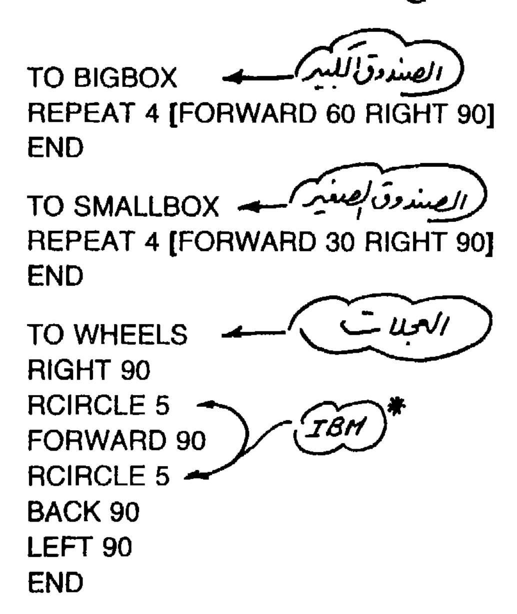


بهذا نحتاج إلى ثلاثة برامج فرعية كالآتى :



```
TO TRUCK
BIGBOX
SMALLBOX
WHEELS
END
```


فلننشىء الآن البرامج الفرعية واحداً تلو الآخر .



```
TO BIGBOX
REPEAT 4 [FORWARD 60 RIGHT 90]
END

TO SMALLBOX
REPEAT 4 [FORWARD 30 RIGHT 90]
END

TO WHEELS
RIGHT 90
RCIRCLE 5
FORWARD 90
RCIRCLE 5
BACK 90
LEFT 90
END
```

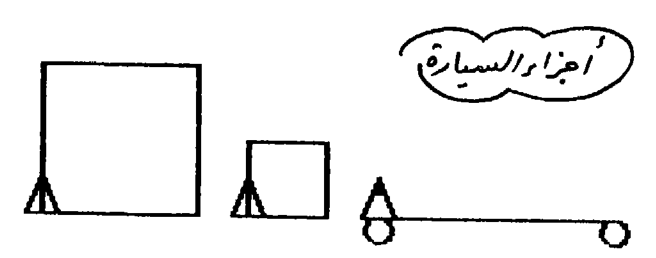
ولا يفوتنا فى البرنامج الفرعى الأخير ظهور العبارات RCIRCLE التى يختص بها الكومبيوتر IBM وعلينا إنشاء ما يناظرها من برامج فرعية لرسم دائرة نصف قطرها 5 عند استخدام كومبيوتر آخر .

والبرنامج الفرعى CIRCLE يصلح لهذا الغرض حيث أنه يرسم دائرة ذات نصف قطر 5 . ومع ذلك فمن المفضل إنشاء برنامج عام لرسم دائرة بأى نصف قطر مطلوب باستخدام الدوال كالمثال الآتى :

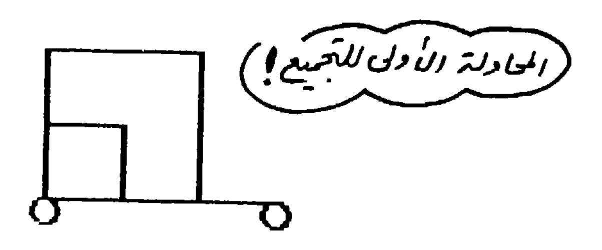
```
?
TO CIRG :R
MAKE "S 360 / :R / 2 / 3.1415927
REPEAT 360 / :S [FD 1 RT :S]
END
```

برنامج فرعى لرسم دائرة نصف قطرها R

ويستخدم هذا البرنامج بكتابة اسمه (CIRG) متبوعاً بنصف القطر المطلوب . وببناء هذه البرامج الفرعية يمكننا الحصول الأشكال التالية التى تتكون منها السيارة .



وإذا نفذنا البرنامج (TRUCK) الآن فسوف نحصل على الشكل التالى :



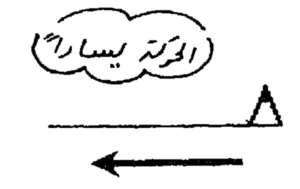
لا تزال السيارة فى حاجة إلى تعديل حتى نحصل على الشكل المطلوب . فيلزمنا تحريك الأجزاء يميناً ويساراً حتى تتم عملية التجميع على الوجه المراد .

وهذا يتطلب بناء برنامجين فرعيين آخرين الأول للحركة يميناً وليكن اسمه MOVEOVER والثانى للحركة يساراً وليكن اسمه MOVEBACK .

```
TO MOVEOVER
RIGHT 90
FORWARD 60
LEFT 90
END
```




```
TO MOVEBACK
LEFT 90
FORWARD 60
RIGHT 90
END
```

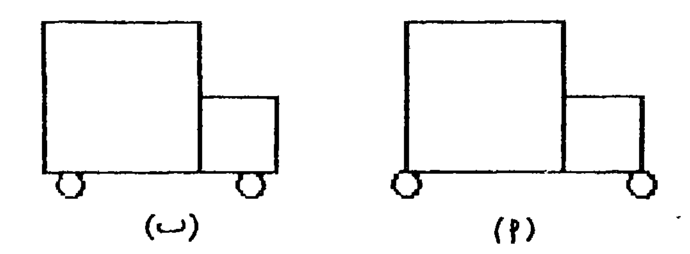


وبإدخال هذه التعديلات على البرنامج الرئيسى يصبح كالتالى :

```
TO TRUCK
BIGBOX
MOVEOVER
SMALLBOX
MOVEBACK
WHEELS
```

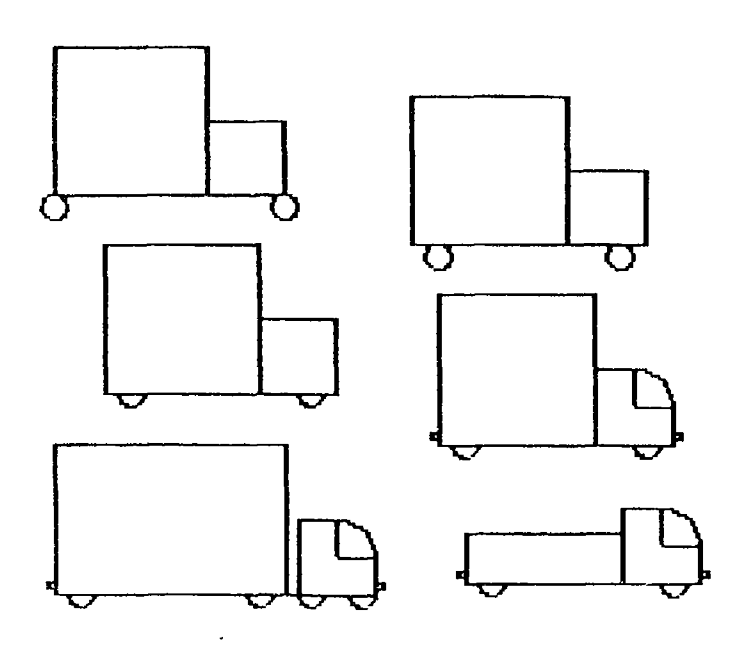
البرنامج الرئيسى

وبتنفيذ البرنامج TRUCK الآن فإنه يعطى الصورة (أ) الموضحة فى الشكل التالى وهى تختلف قليلاً عن الشكل الأصلى الذى رسمناه باليد من ناحية مكان العجلات . فإذا أردت الحصول (ب) المطابقة للشكل الأصلى فعليك بإجراء التعديل اللازم بالبرنامج .



هذه السيارة هى الخلية الأولى التى نبنى منها قافلة السيارات سواء بالتكرار أو بإضافة التعديلات إلى شكل السيارة لكى تشتمل القافلة على سيارات النقل « ونصف النقل » بمختلف الأحجام والأشكال .

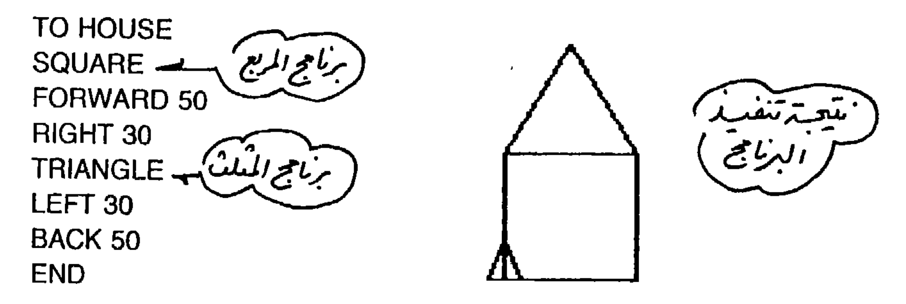
والشكل التالى يوضح بعض هذه النماذج التىَ يمكن برمجتها بسهولة .




SQUARE
GROW SQUARE
SIZE
40
30
SQUARE

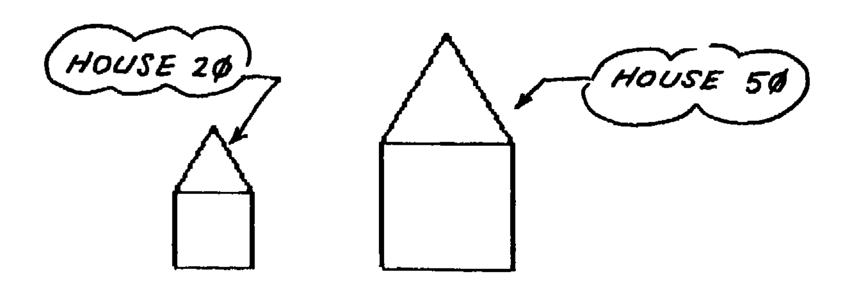
● منطقة سكنية :

لنبدأ بمنزل واحد بالطريقة التقليدية . والمنزل يتكون من مربع ومثلث . وكل من المربع والمثلث يمكن برمجته على حدة كبرنامج فرعى صغير باستخدام ضلع طوله ٥٠ خطوة .



ويمكن التحكم فى حجم المنزل الناتج باستخدام متغير SIZE لتحديد حجم المربع والمثلث معاً . والمتغير SIZE يمثل طول الضلع لأى من الشكلين فيصبح البرنامج كالآتى :

```
TO HOUSE :SIZE
SQUARE :SIZE
FORWARD :SIZE
RIGHT 30
TRIANGLE :SIZE
LEFT 30
BACK :SIZE
END
```



وكما نرى أنه يمكن الحصول على أحجام مختلفة من المنازل بتحويل البرنامج إلى برنامج للدالة بدلاً من الطريقة العادية والرسم يوضح منزلين أحدهما تم تنفيذه بالأمر .

? HOUSE 20

والآخر باستخدام الأمر :

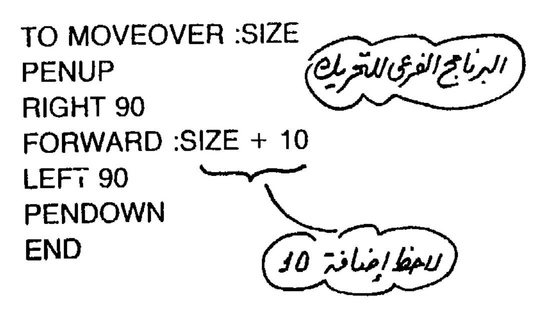
? HOUSE 50

وللتوسع فى بناء المنازل يلزمنا روتين لتحريك الشكل يميناً أو يساراً على غرار الروتين MOVEOVER أو الروتين MOVEBACK .

ولتعميم الروتين يمكن إضافة تعديلين إلى البرنامج MOVEOVER .

أولاً : استخدام الدوال فى برنامج تحريك الأشكال حتى يكون عاماً بحيث نستخدمه فى أى برنامج .

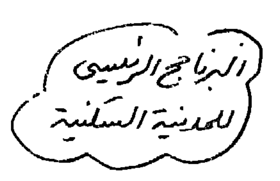
ثانياً : نكتفى باستخدام روتين واحد وليكن MOVEOVER بحيث يستقبل قيماً موجبة للمتغير عند الحركة يميناً وقيماً سالبة عند الحركة إلى اليسار . وهذا هو البرنامج الفرعى :

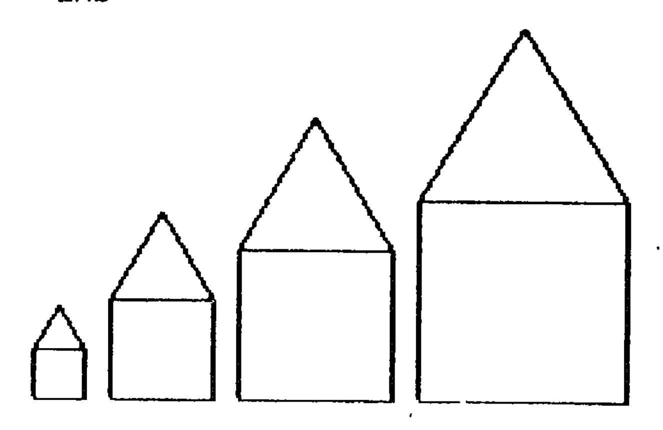

```
TO MOVEOVER :SIZE
PENUP
RIGHT 90
FORWARD :SIZE + 10
LEFT 90
PENDOWN
END
```


ولا يفوتنا ملاحظة إضافة 10 على المتغير SIZE مما ينتج عنه « انحياز » إلى اليمين قدره عشر خطوات . فإذا أدخلنا الرقم 60 للبرنامج تتم الإزاحة بمقدار 70 ، وإذا أدخلنا الرقم 60ــ تتم الإزاحة بمقدار (50 −) .

أما برنامج المساكن فهو كالآتى :

```
TO HOUSES
MOVEOVER -100
HOUSE 10
MOVEOVER 10
HOUSE 20
MOVEOVER 20
HOUSE 30
MOVEOVER 30
HOUSE 40
MOVEOVER 40
END
```





وهناك مع ذلك طريقة بديلة مختصرة وهى بناء البرنامج الرئيسى نفسه كدالة باستخدام متغير SIZE أيضاً كالآتى :

```
TO GROWHOUSES :SIZE
HOUSE :SIZE
MOVEOVER :SIZE
GROWHOUSES :SIZE + 10
END
```

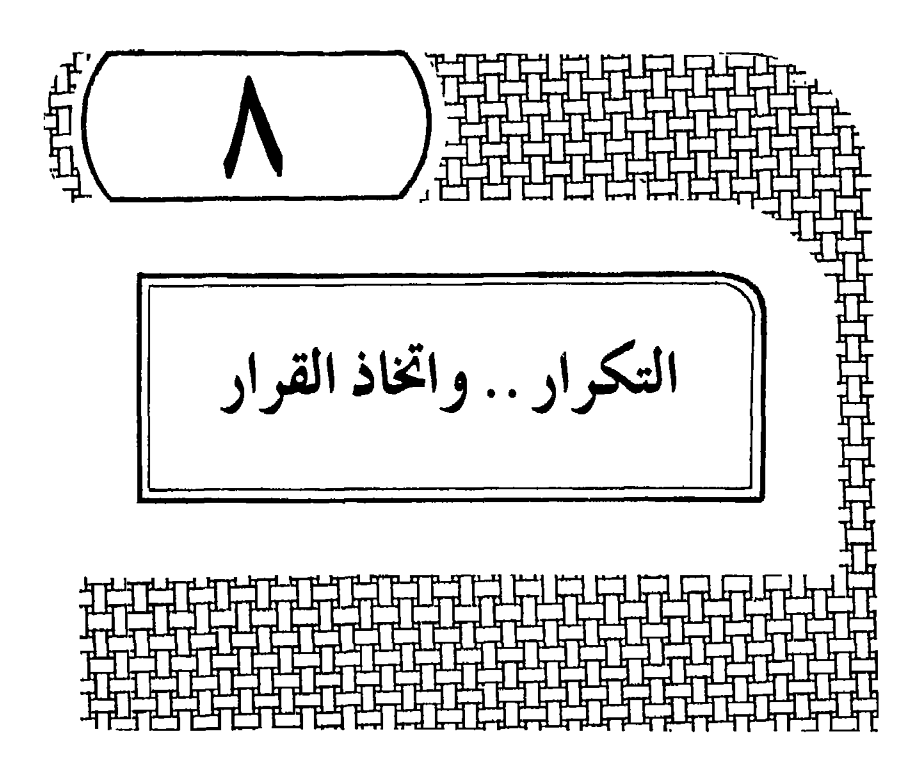
التكرار مع زيادة الحجم

وهذا البرنامج يزيد حجم المنزل كل مرة بإضافة 10 إلى المتغيّر SIZE .

والصفة الجديدة التى يختص بها هذا البرنامج عن سابقة أنه لن يتوقف إلاّ إذا تدخلت من الخارج بالضغط على الزر (ctrl-beak) أو ما يعادله فى الأجهزة المختلفة .

ولو تركت البرنامج يعمل لفترة طويلة فإنك تحصل على نتيجة مؤسفة إذ تتراكم المنازل فوق بعضها البعض حتى تسوّد الشاشة تماماً من كثرة الخطوط المتزاحمة بها .

وسنرى فى الأجزاء التالية كيف نوقف مثل هذا البرنامج من داخل البرنامج نفسه .



٨

التكرار .. واتخاذ القرار

• الحلقات التكرارية :

أحياناً يدور الإنسان فى حلقة مفرغة لا يستطيع الخلاص منها إلاّ بمعجزة أو بتدخل حاسم من الأهل أو الأصدقاء المخلصين .

وهذا الموقف كثيراً ما يتعرض له الساحر لوجو عند تنفيذه لأمر ما فيجد نفسه واقعاً فى حلقة تكرارية بلا نهاية لا يستطيع الخروج منها .

وقد تصادف السلحفاه أيضاً هذا الموقف وهى ترسم على الشاشة دون أن ينتهى عملها كما حدث فى برنامج المنطقة السكنية .

ولا بد فى هذه الحالة من الاستعانة « بنظام التشغيل » للكومبيوتر الذى يسعفنا بالضغط على زر معيّن فيوقف أى برنامج جارى تنفيذه ..

والمثال الآتى يوضح نوعية من هذه الحلقات التكرارية اللانهائية حيث يصاب أحد الروتينات الفرعية (LOOP) بنوبة من جنون العظمة فيستمر فى القول « أنا رجل عظيم .. I am a great man » بلا نهاية . ولا يتوقف هذا البرنامج الفرعى (LOOP) إلاّ عندما نضغط على زر الإيقاف الاضطرارى فيتوقف مُجبراً ..



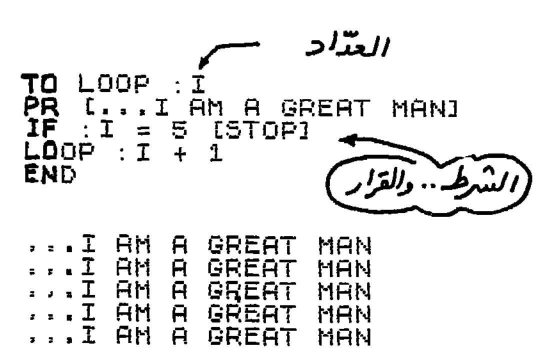
● كيف نوقف الحلقة التكرارية IF , STOP

لعلنا لاحظنا فى البرنامج السابق أن البرنامج لا يصل إلى نهايته أبداً فكلما طبع القائمة I am a great man استدعى البرنامج نفسه من جديد بالأمر LOOP فيكرر ما يفعل .

ولكى نوقف مثل هذا البرنامج لا بد من أن نضمنه خاصية القدرة على اتخاذ القرار . ولنفرض أننا نريد تكرار الحلقة خمس مرات فقط . هذا معناه أن يجب أن يحتوى البرنامج على عدّاد ليعّد من واحد إلى خمسة وبجانب ذلك يجب أن يكون هناك من يراقب العدّاد فإذا وصل إلى الرقم خمسة تدخل لإنهاء العمل وإيقاف البرنامج .

لا شك أننا سنحتاج لتعلم بعض الأوامر الجديدة من أوامر لوجو .

أنظر هذا التعديل فى البرنامج :



لقد استخدمنا الدليل I فى مستهل البرنامج كعدّاد .

وبعد طبع كل قائمة يزداد العداد بمقدار واحد كما فى السطر الأخير :

LOOP :I + 1

أى أن البرنامج عندما يستدعى نفسه للتكرار فإنه يزيد قيمة الدليل بمقدار 1- . هذا هو العدّاد ..

أما القرار فيأتى بعد طبع القائمة مباشرة بموجب العبارة الجديدة :

IF :I = 5 [STOP]

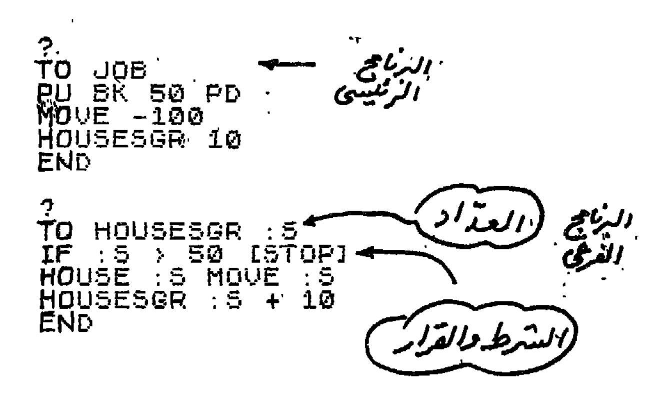
تقول هذه العبارة : « إذا كان العدّاد I مساوياً للرقم 5 فنفذ الأمر : STOP . وإلاّ فانتقل إلى الأمر التالى للأمر STOP فإذا تحقق الشرط فى الدورة الخامسة نفذ البرنامج الأمر STOP وخرج من الحلقة . ويمكن فهم العملية الشرطية بصورة عامة كالآتى :

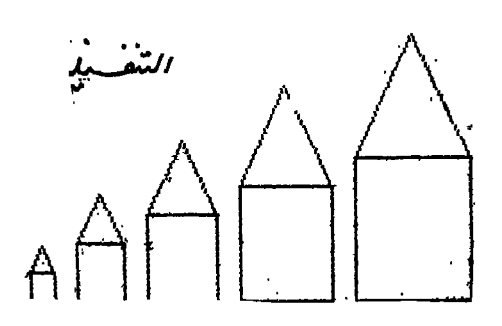
القائمة الثانية [القائمة الأولى] الشرط IF

فإذا تحقق الشرط أياً كان هذا الشرط فإن قائمة الأوامر يتم تنفيذها أما إذا لم يتحقق الشرط فإن التنفيذ يتجه إلى القائمة الثانية . ونلاحظ أن القائمة الأولى توضع بين القوسين المربعين لفصلها عما يليها من أوامر .

والقائمة الأولى هنا تحتوى على أمر واحد هو STOP ولكن يجوز أن تشمل أكثر من أمر .

● ويمكن وضع شرط أيضاً فى برنامج المساكن بنفس الطريقة فلو أردنا مثلاً ألاّ يزيد حجم أكبر منزل عن ٥٠ فإن الشرط يمكن أن يتحكم فى الحجم SIZE كالآتى :





والبرنامج الذى يقوم ببناء المنازل هو البرنامج الفرعى HOUSESGR :S والمتغير S هو نفسه متغير الحجم SIZE حيث يتحكم فى حجم المنازل بزيادته بمقدار 10 عند كل رسم جديد .

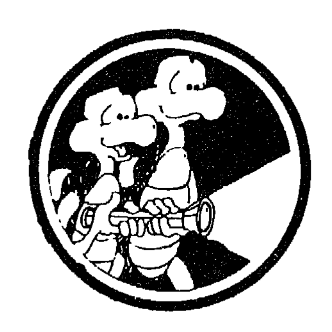
أما البرامج الفرعية المستخدمة بداخل هذا البرنامج فهى MOVE وهو ذاته البرنامج MOVEOVER الذى استخدمناه من قبل .

والشرط الذى وضعناه فى البرنامج يقول :

« إذا زاد حجم المنزل عن 50 توقف »

أما البرنامج الرئيسى JOB فهو يمهد للبرنامج HOUSESGR بتحريك السلحفاه إلى مكان بداية مناسب باستخدام البرنامج الفرعى MOVE .

فـلاش



ما الفرق بين التكرار باستخدام الأمر REPEAT وبين استخدام الحلقات التكرارية ذات الشرط التى تحدثنا عنها مؤخراً ؟

إن الفارق الرئيسى هو أن الأمر REPEAT يفيد فى تكرار عملية معينة عدداً معيناً من المرّات .

أما استخدام الشرط فهو أكثر عمومية حيث يمكن أن يتحكم الشرط فى عدد المرات ، أو فى أى بيان آخر بصرف النظر عن عدد المرات . وفى المثال الأخير يتحكم الشرط فى حجم المنزل . وفى الأمثلة القادمة سوف نلتقى بالعملية الشرطية وهى تتحكم فى بيانات مختلفة تماماً بدون تحديد مسبق لعدد المرّات .

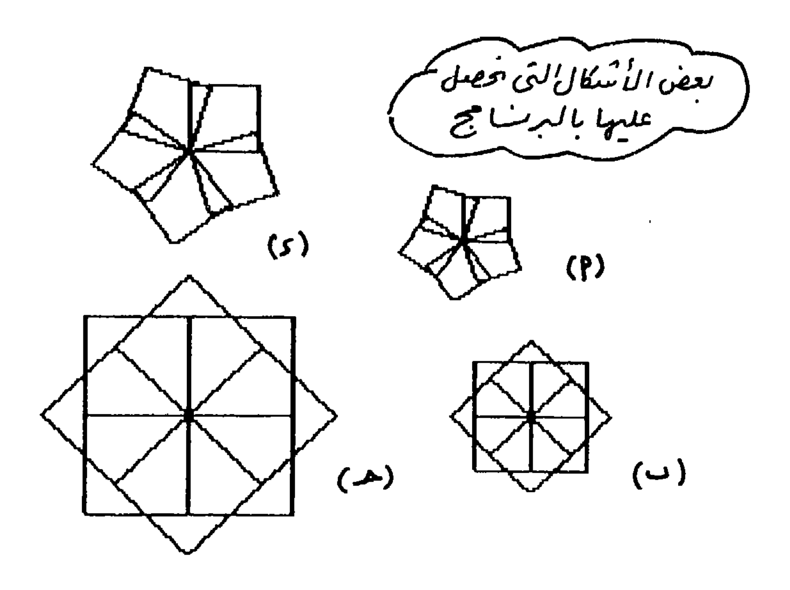
• التحكم فى وضع السلحفاه HEADING

اكتب هذا البرنامج الذى يستدعى البرنامج الفرعى SQUARE (وهو برنامج رسم المربع) ويديره بزاوية معينة خلال حلقة تكرارية لا نهائية .

وهو يعمل بإدخال رقمى طول الضلع والزاوية مثل :

? SPINSQUARES2 50 45

```
TO SPINSQUARES2 :SIZE :ANGLE
SQUARE :SIZE
RIGHT :ANGLE
SPINSQUARES2 :SIZE :ANGLE
END
```



مثل هذا البرنامج يصعب إيقافه بتحديد عدد مرات التكرار للمربّع الدائر من ناحية لأن هذا العدد يتغير مع كل زاوية مُدخلة ، فمع الزاوية ٩٠ درجة نحصل على ٤ مربعات ومع الزاوية ٤٥ درجة ثمانية مربعات كما فى الشكل (جـ) ومن ناحية أخرى فإنه من الصعب التنبؤ مع بعض الزوايا بعدد مرات التكرار اللازمة لرسم الشكل النهائى المتماثل .

لكن هناك صفة مشتركة فى جميع الأشكال التى نحصل عليها من البرنامج وهى أن الشكل يكتمل عندما تعود السلحفاه إلى وضعها الابتدائى . وهذا هو الجانب الذى يمكننا التحكم فيه . فعند الوضع الابتدائى تكون السلحفاه متجهة إلى أعلى ويمكن وصف ذلك بأمر لوجو جديد هو HEADING بمعنى اتجاه رأس السلحفاه وهو يقاس بالدرجات . فعند الوضع الابتدائى للسلحفاه

يكون اتجاه الرأس صفراً . وإذا أدخلت الأمر 90 RT يصبح الاتجاه 90 درجة . ويمكنك التأكد من ذلك بتجربة بعض الأوامر مثل :

? RT 90	لإدارة السلحفاه
? PR HEADING	ثم .. لطبع اتجاه الرأس
90 ←	الإجابة ....
? RT 20	وإدارتها مرة أخرى
? PR HEADING	ثم .. طبع الاتجاه الجديد
110	الإجابة

وهكذا

يمكننا الآن إضافة الشرط التالى فى البرنامج :

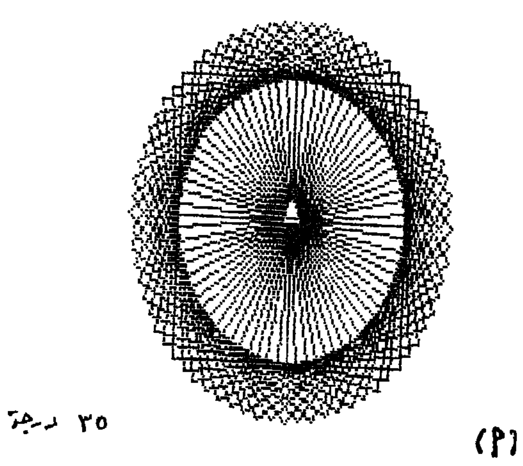
IF HEADING = 0 [STOP]

أنظر الصورة الجديدة للبرنامج .

```
TO SPINSQUARES3 :SIZE :ANGLE
SQUARE :SIZE
RIGHT :ANGLE
IF HEADING = 0 [STOP]   ← الشرط
SPINSQUARES3 :SIZE :ANGLE
END
```

وبذلك يتوقف البرنامج عندما يتم رسم الشكل الكامل مرّة واحدة .

وتظهر أهمية هذا الشرط مع بعض الزوايا مثل ٣٥ درجة و ١٥ درجة وهما تعطيان الشكلين أ ، ب الموضحين بعد بالترتيب .



٣٥ درجة (أ)

١٥ درجة (ب)

● ماذا يحدث بداخل الكومبيوتر ؟

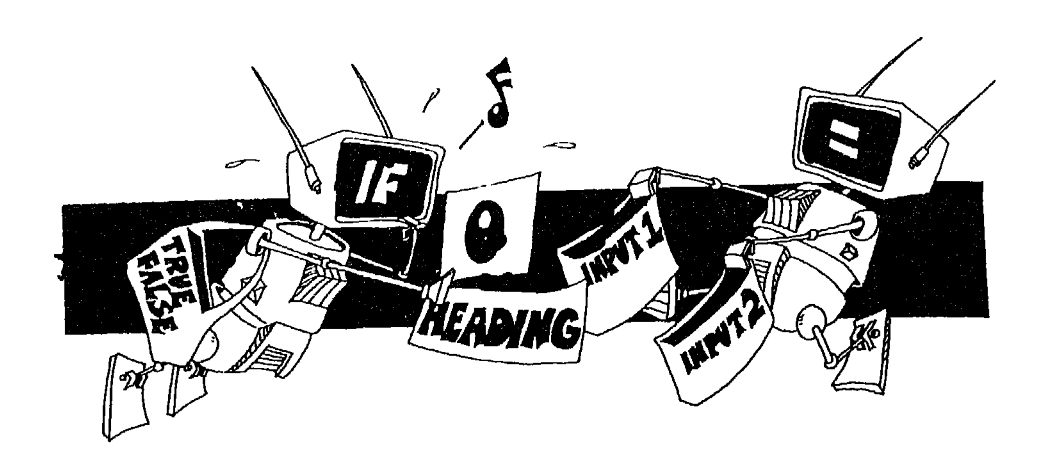
تتبع هذه القصة المصورة الطريفة التى تحكى عن دور العناصر المكونة للعملية الشرطية الآتية :

IF HEADING = 0 [STOP]

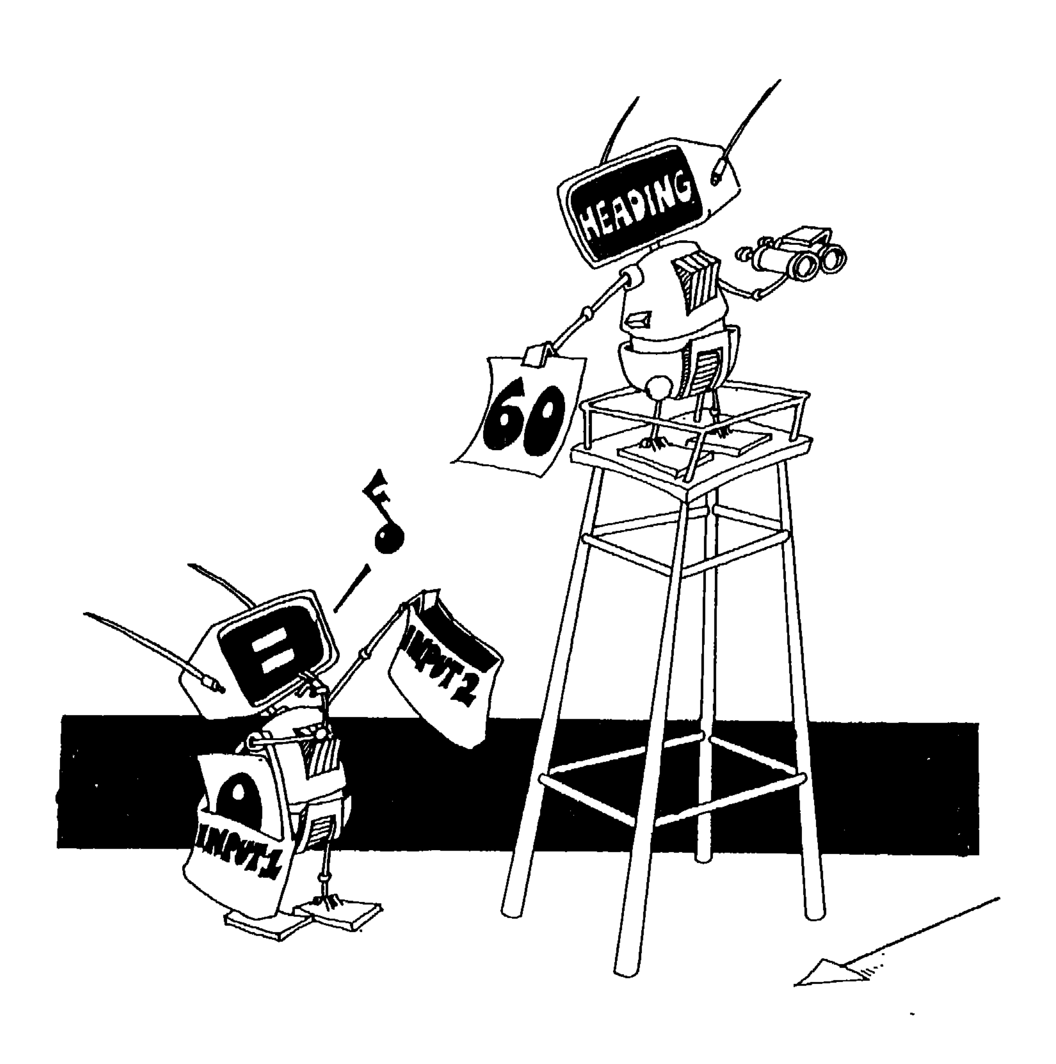
وهذه العناصر هى :

STOP ، = ، HEADING ، IF

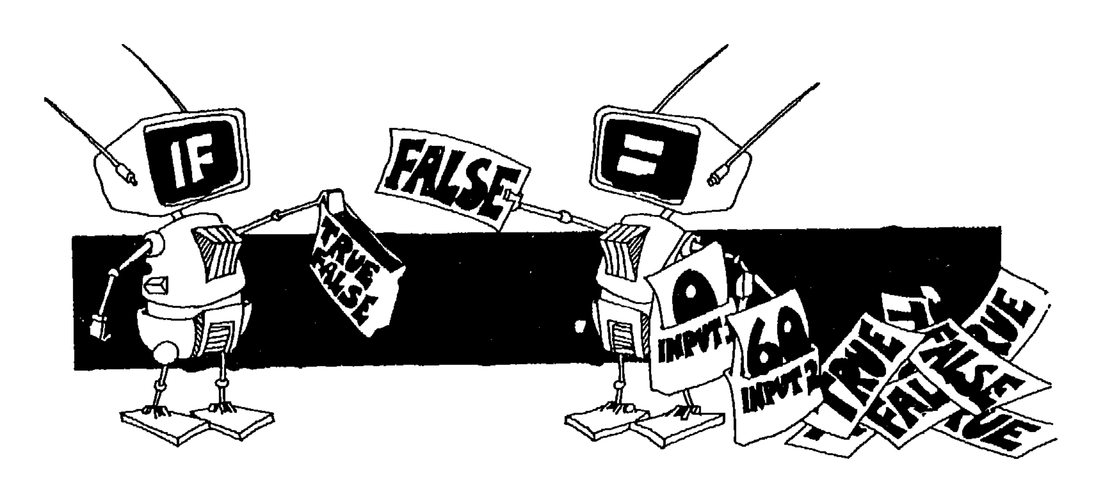
وكلها تسمى أوامر لوجو الأولية (primitives) لذلك نمثلها بالروبوط ذى الشاشة المربعة .



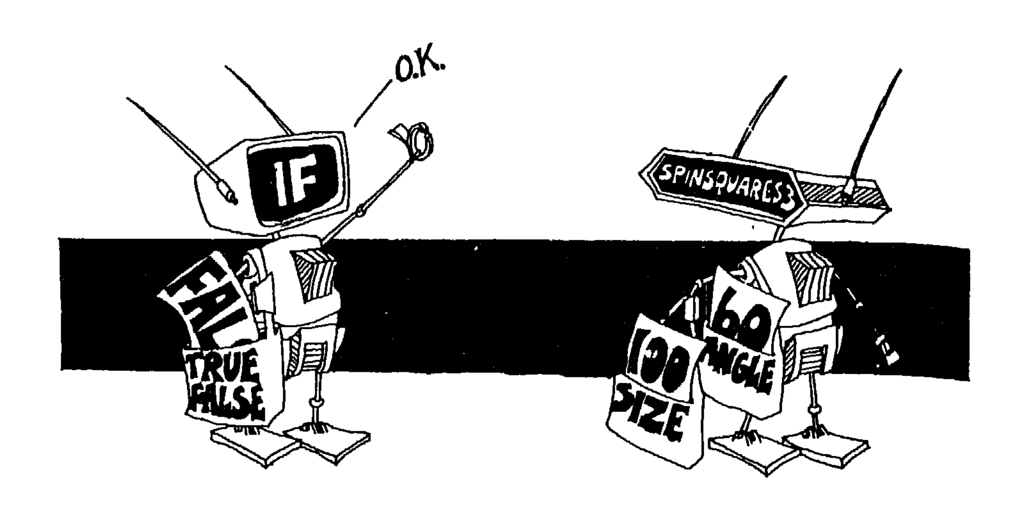
(أ) الروبوط IF يسلم الروبوط (=) مُدخلين هما HEADING ، الصفر . ثم ينتظر منه أحد نتيجتين TRUE أو FALSE .



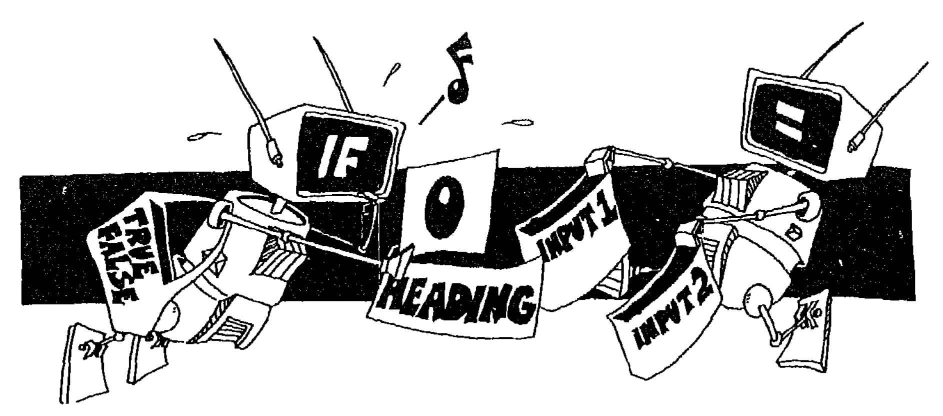
(ب) يقوم الروبوط HEADING بإيجاد قيمة وضع رأس السلحفاه ويسلمه للروبوط (=) ليقارن بينهما .



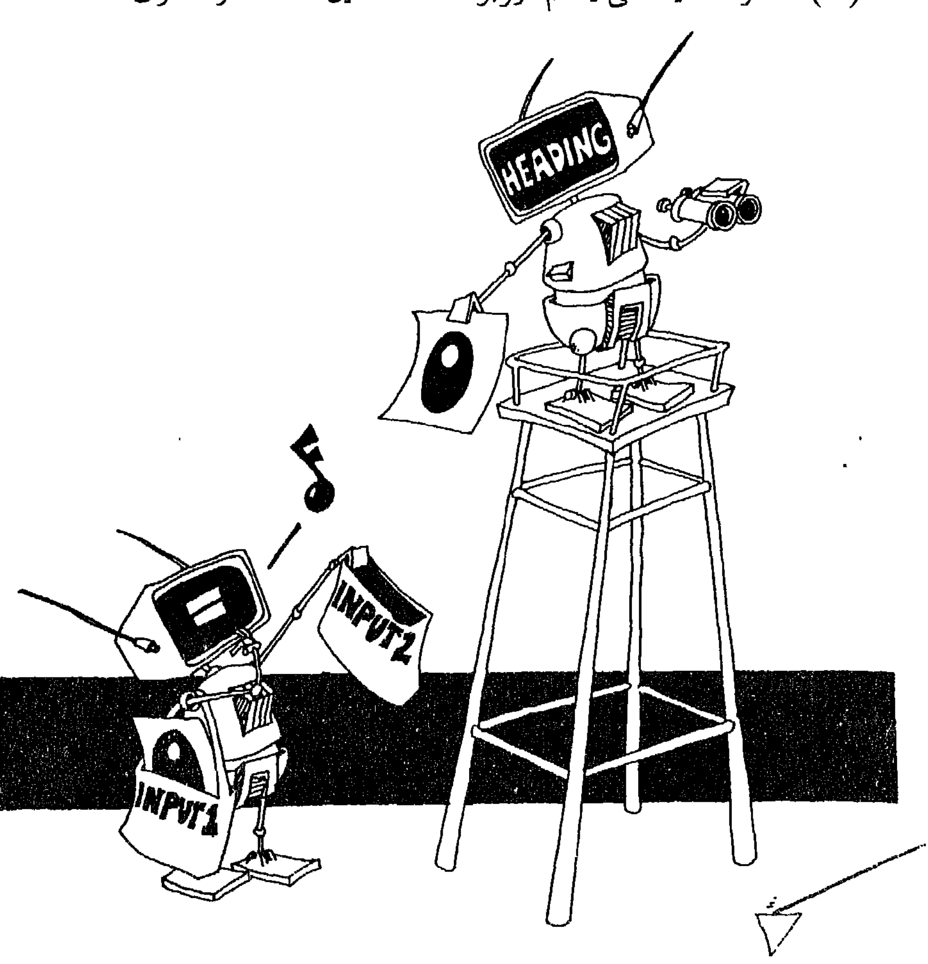
(جـ) الروبوط (=) يرسل نتيجة المقارنة إلى الروبوط (IF) وهى غير صحيحة بالطبع FALSE .



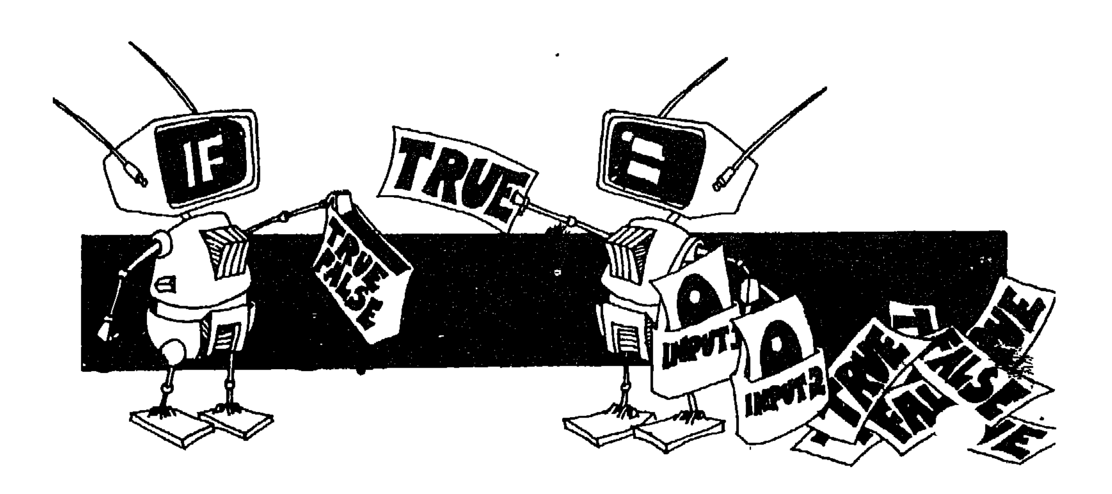
(د) لو كانت نتيجة الاختبار صحيحة TRUE لأمر الروبوط IF بتنفيذ القائمة الأولى [STOP] ولكن نظراً لأن النتيجة هو FALSE فقد أشار الروبوط IF للبرنامج بالاستمرار دون الالتفات إلى القائمة [STOP] ..



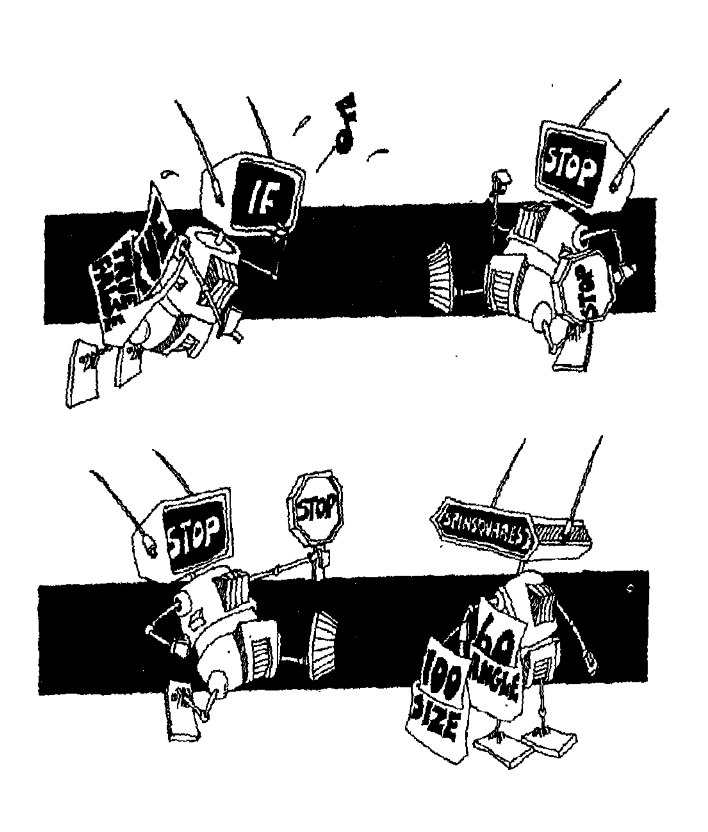
(هـ) تستمر العملية حتى يسلّم الروبوط IF المدخلين IF ، 0 مرة أخرى ..



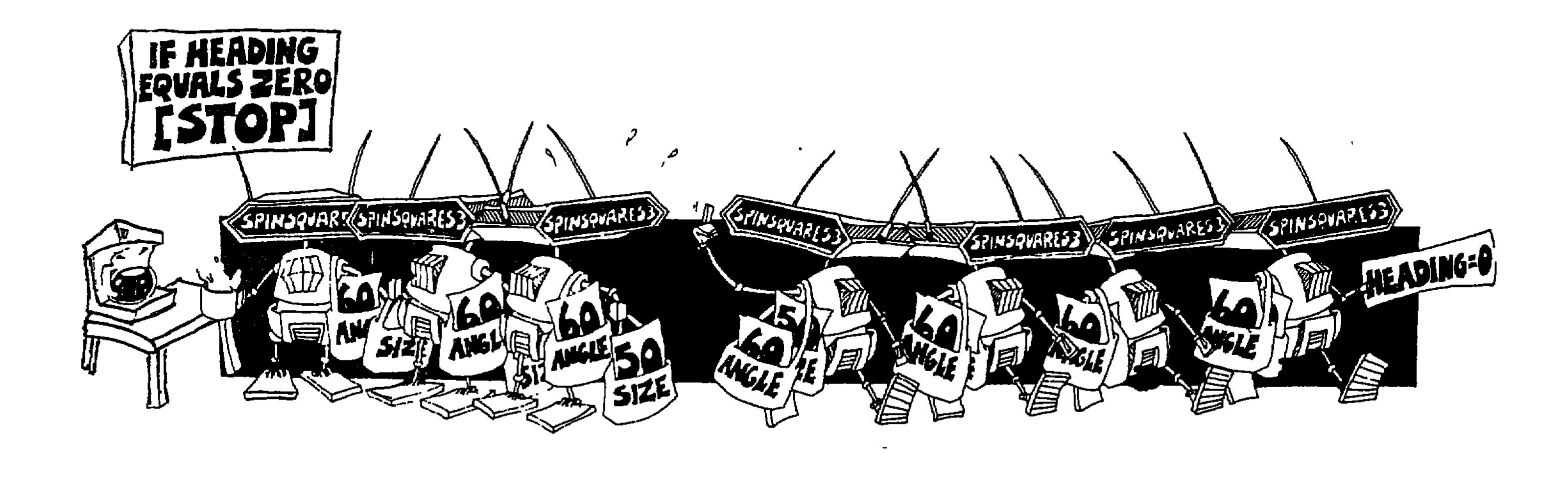
(و) بالبحث والتقصّى عن وضع السلحفاه هذه المرّة وجد الروبوط HEADING أن رأس السلحفاه يشير إلى أعلى فقام بتسليم الرقم صفر للروبوط (=) .



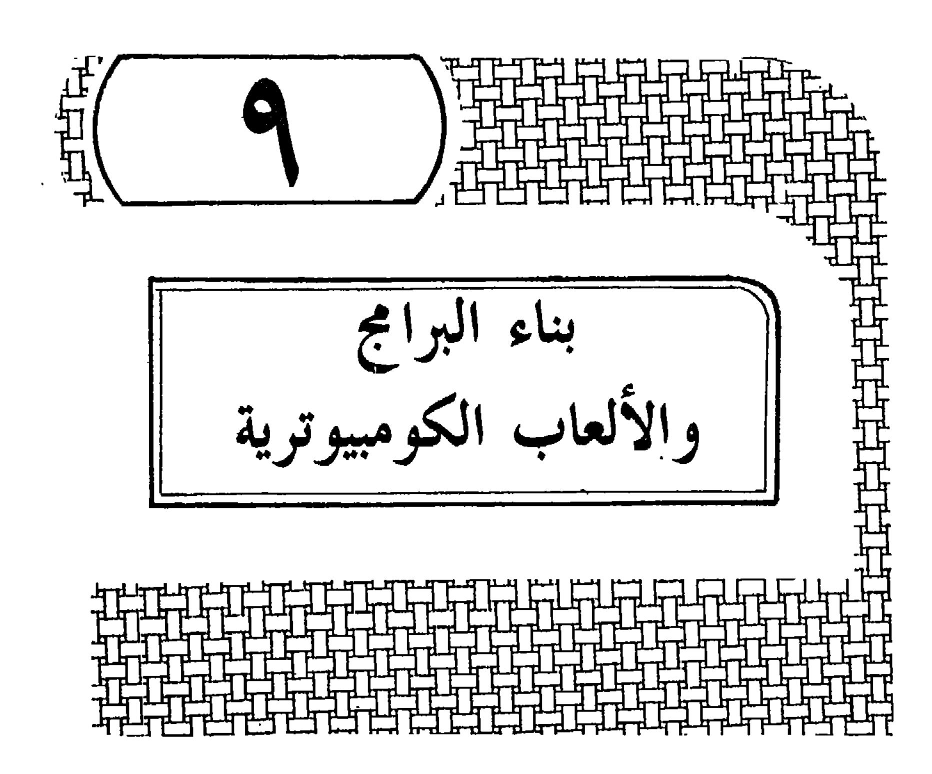
(ز) يقوم الروبوط (=) بإرسال نتيجة الاختبار TRUE إلى الروبوط IF الذى كلّفه بالمهمة .



(ح) بمجرد أن يتسلم الروبوط IF نتيجة الاختبار TRUE يستدعى فوراً الروبوط STOP الذى يقوم بدوره بإيقاف البرنامج .



(ط) إذا كان هناك برنامج فرعية يستدعى بعضها البعض فإنها جميعاً تنتظر الأمر STOP حتى تنهى عملها .. بالدور من الأصغر إلى إلى الأكبر (أو من الفرعى لرئيسى) .



٩

بناء البرامج
والألعاب الكومبيوترية

• تغيير وظائف الأزرار READCHAR (RC)

عند الضغط على أى زر من أزرار الكتابة فإن لبنة واحدة (حرف أو رقم أو علامة خاصة) تُكتب على الشاشة .

ويمكن قراءة اللبنة المُدخلة بواسطة العملية READ CHAR وهى تماثل العملية READLIST ولكنها عملية تختص بقراءة لبنة مفردة ولذلك فبمجرد لمس أحد الأزرار تُنفذ العملية ويتم الانتقال إلى الأمر التالى . ويختصر هذا الأمر إلى RC .

وهذه العملية تفتح الباب إمامنا لتطبيقات كثيرة . فالمثال الآتى يستقبل اللبنة من الزر المضغوط ويخصصها للمتغير Z ثم يستخدمها لأداء وظيفة مخالفة تماماً للوظيفة الأصلية . فهو يستخدم الرقم 5 فى إدارة السلحفاه إلى اليسار بمقدار ٩٠ درجة ويستخدم الرقم 6 لتحريك السلحفاه إلى الخلف بمقدار ١٠ خطوات .. وهكذا .. وبذلك نحصل على وسيلة مباشرة للتحكم فى السلحفاه بالأزرار بدلاً من التحكم فيها من خلال برنامج واحد ثابت .

```
TO  DRAW
MAKE  "Z  RC
IF  :Z = 5  [ LT  90 ]
IF  :Z = 6  [ BK  10 ]
IF  :Z = 7  [ FD  10 ]
IF  :Z = 8  [ RT  90 ]
DRAW
END
```

• الكلمات والقوائم كمتغيرات :

كما تعاملنا من قبل مع متغيرات الأعداد فإنه يمكن حفظ القوائم والكلمات أيضاً فى متغيرات . والمثال الآتى يوضح إحدى الطرق للتخصيص .

```
TO SPEAK :MESSAGE
PRINT [THE MESSAGE I AM GOING TO PRINT IS]
PRINT :MESSAGE
END
```

يستقبل هذا البرنامج الرسالة MESSAGE منك ثم يطبعها . والرسالة قد تكون كلمة أو قائمة أو رقما ولكن يراعى طريقة إدخال كل نوعية .

فإذا كنت تعتزم إدخال رقم للبرنامج فأنت تكتب اسم البرنامج متبوعاً بالرقم المُدخل كالآتى :

SPEAK 144

أما إذا كنت تنوى إدخال كلمة للمتغير MESSAGE فيتم ذلك كالآتى :

SPEAK "HELLO

فإذا كان المُدخل عبارة عن قائمة فإننا نستخدم علامة القائمة كالآتى :

SPEAK [GOOD MORNING SIR]

وهذه الطريقة تماثل تماماً ما اتبعناه مع الأمر MAKE الذى استخدمناه من قبل للتخصيص للمتغيرات سواء مع الأعداد أو الكلمات أو القوائم .

وللتذكرة نقدم المثال الآتى :

```
MAKE "MESSAGE [THIS IS GETTING SILLY]
```

ولعلنا نذكر أن المتغير المستخدم مع MAKE لا بد أن يكون كلمة أى أنه يبدأ بالعلامة (") أما البيان فيمكن أن يكون كلمة أو قائمة أو عدداً . بهذا الأمر السابق يتم تخصيص القائمة التى بين القوسين للمتغير MESSAGE .

إن المتغيرات التى تحتوى على كلمات وقوائم مثل المتغيرات التى تحتوى على

أعداد فهى قد تكون متغيرات لبرامج الدوال كما فى المثال الأول ، أو متغيرات عامة كما فى المثال الثانى .

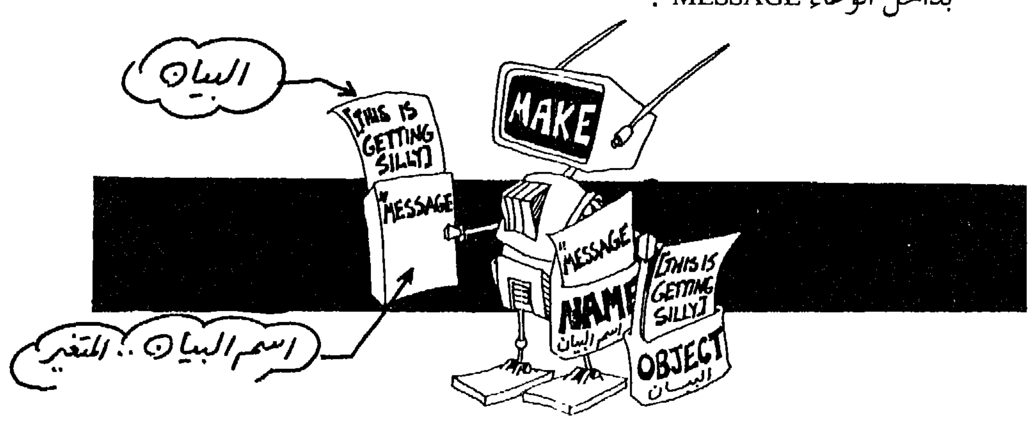
ولعلنا نتذكر طريقتى تخزين كل من المتغيرات العامة ومتغيرات برامج الدوال والفارق بين الطريقتين .

فالمتغيرات العامة تُعتبر مشاعاً فى الذاكرة أما متغيرات الدوال فتستخدم داخل برنامجها فقط .

ولعل الساحر لوجو يشرح لنا هذا الأمر بطريقة أفضل .

● ماذا يحدث بداخل الكومبيوتر :

عادة يحمل الروبوط MAKE حقيبتين واحدة للبيانات وواحدة لأسماء البيانات (المتغيرات) . لذلك فعند إعطاء الأمر الوارد بالمثال الأخير فإن الروبوط MAKE يقوم بوضع القائمة [THIS IS GETTING SILLY] بداخل الوعاء MESSAGE :



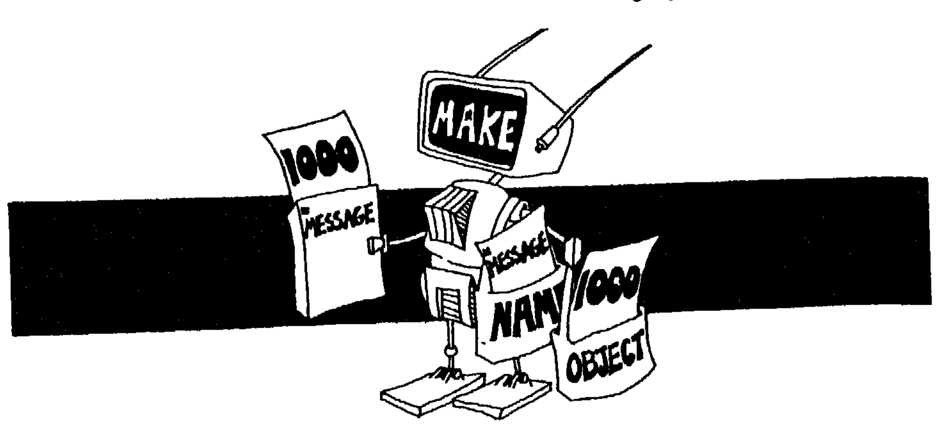
.. ويقوم الساحر لوجو باستلام العلبة MESSAGE بما فيها ليخزنها فى ذاكرة الكومبيوتر الرئيسية تحت الطلب .



والوعاء MESSAGE يمكن أن نستبدل ما فيه من كلمات بأرقام أيضاً وبمجرد إعطاء أمر MAKE جديد .

MAKE "MESSAGE 1000

فيقوم على الفور الروبوط MAKE بوضعها فى الوعاء MESSAGE تمهيداً لتسليمه إلى الساحر لوجو .

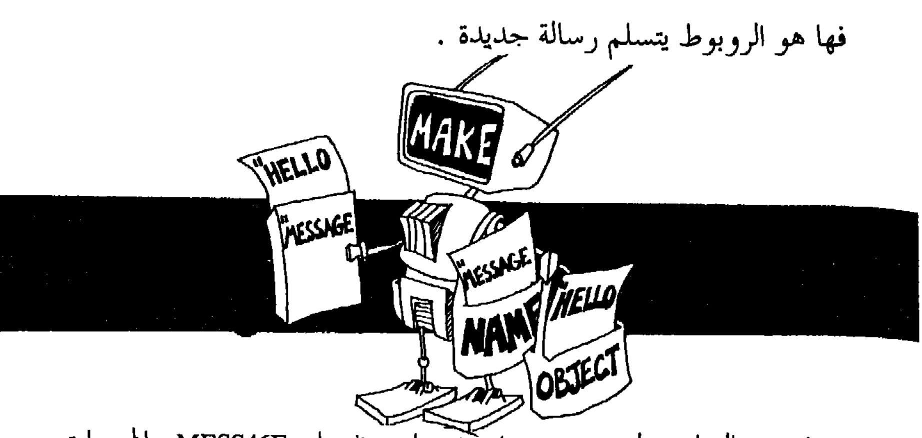




ومحتويات المتغير MESSAGE يمكن تغييرها فى أى وقت بأمر MAKE جديد فيبدأ الروبوط MAKE والساحر لوجو بإعادة العمل مرة أخرى . وها هما ينفذان الأمر :

MAKE "MESSAGE "HELLO

فها هو الروبوط يتسلم رسالة جديدة .



وها هو الساحر لوجو يستبدل محتويات الوعاء MESS46E بالمحتويات الجديدة **"HELLO** ..



بهذا يصبح وعاء المتغير MESSAGE محتوياً على العدد 1000 . أما البيانات القديمة فذهبت إلى غير رجعة . وللتأكد يمكنك إعطاء الأمر :

PRINT :MESSAGE

سوف نجد الرد جاهزاً وهو 1000 .

بل يمكنك إعطاء هذا الأمر عقب كل تغيير في البيان بالأمر MAKE حتى تشاهد كيف تتبدل البيانات في الوعاء MESSAGE .

● جمع الأعداد ووصل القوائم والكلمات :

صادفنا من قبل أمثلة كالآتي :

```
MAKE "NUMBER 5
PRINT :NUMBER
MAKE "NUMBER :NUMBER + 5
PRINT :NUMBER
```

(أضف 5 إلى قيمة المتغير)

فالمتغير NUMBER الذى يحتوى على العدد 5 يمكننا أن نضيف إليه (أو نطرح منه ..) ما شئنا من الأرقام دون تغيير اسمه . وقد استخدمنا ذلك من قبل فى إنشاء العدّادات .

المتغيرات التى تحتوى على قوائم وكلمات يمكن أيضاً معالجتها بطريقة مماثلة ولكن عندما نتعامل مع البيانات غير العددية فإننا نطلق على عملية الجمع إسماً جديداً هو عملية الوصل (catenation) .

ولغة لوجو تمدنا بوسيلة جاهزة للوصل بين القوائم هى SENTENCE . ولنر معاً هذا البرنامج المشابه للبرنامج السابق .

```
MAKE "MESSAGE [HELLO THERE]
PRINT :MESSAGE
MAKE "MESSAGE SENTENCE :MESSAGE "FRIEND
PRINT :MESSAGE
```

صل المتغير MESSAGE بكلمة FRIEND

واسم المتغير فى حالتنا هذه هو MESSAGE وهو يحتوى على القائمة :

[HELLO THERE]

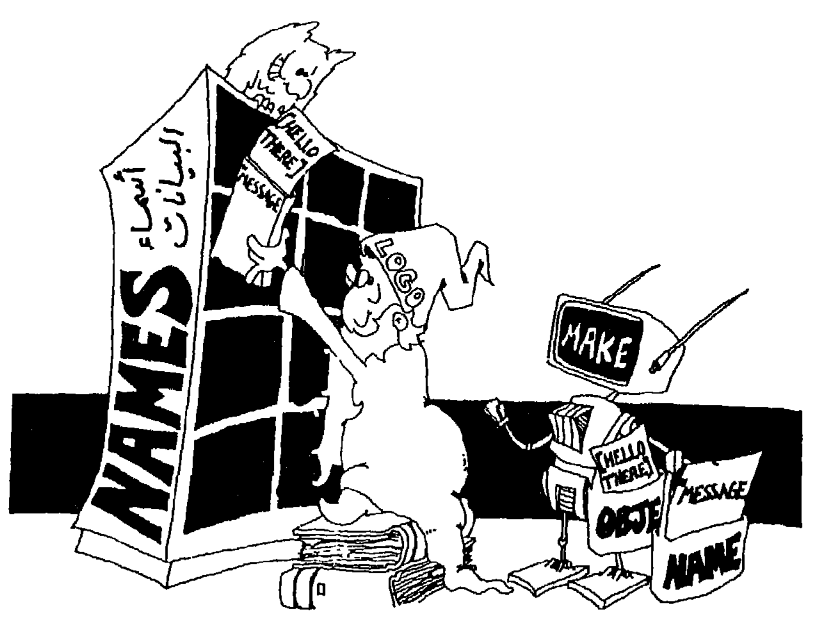
وفى السطر الثالث من هذا البرنامج نجرى عملية الوصل على كلمة FRIEND المميزة بعلامة الاقتباس المفردة قبلها (") مع قيمة المتغير MESSAGE المميزة بالنقطتين (:) مع استخدام الأمر SENTENCE للوصل بين الكلمة والقائمة .

لذلك عن تشغيل هذا البرنامج نحصل على العبارة الآتية :

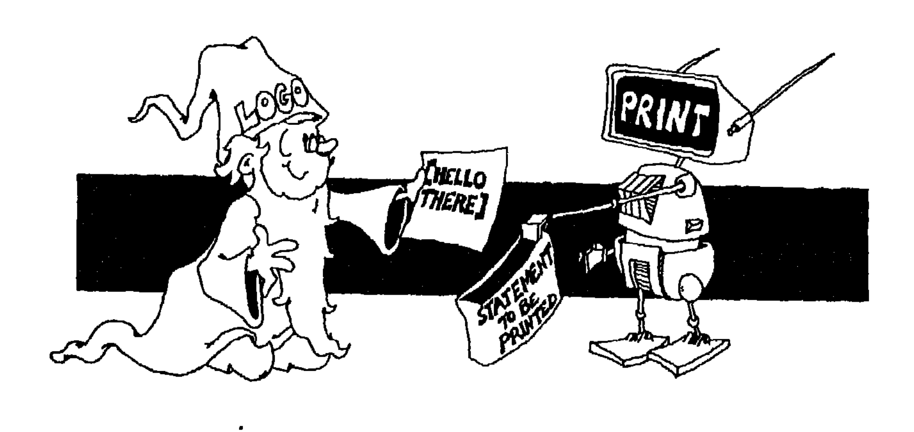
HELLO THERE FRIEND

● ماذا يحدث بداخل الكومبيوتر

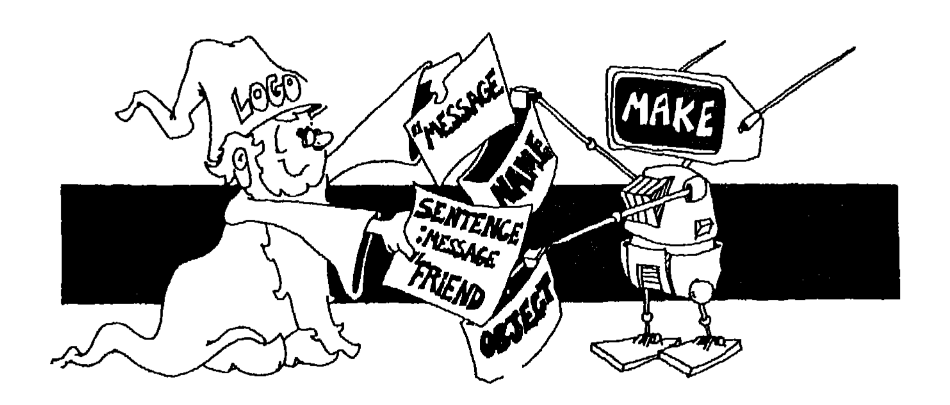
ولنر الآن كيف يقوم الساحر لوجو بهذه العملية :



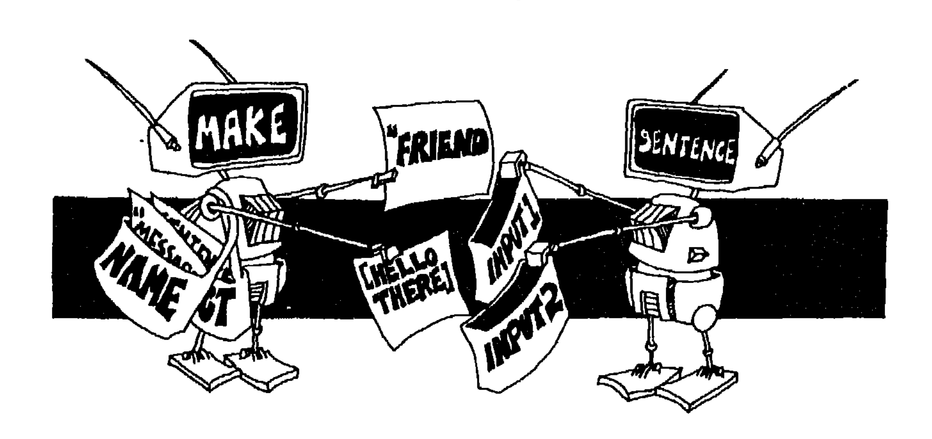
ها هو الساحر لوجو يضع علبة المتغير MESSAGE فى الذاكرة .



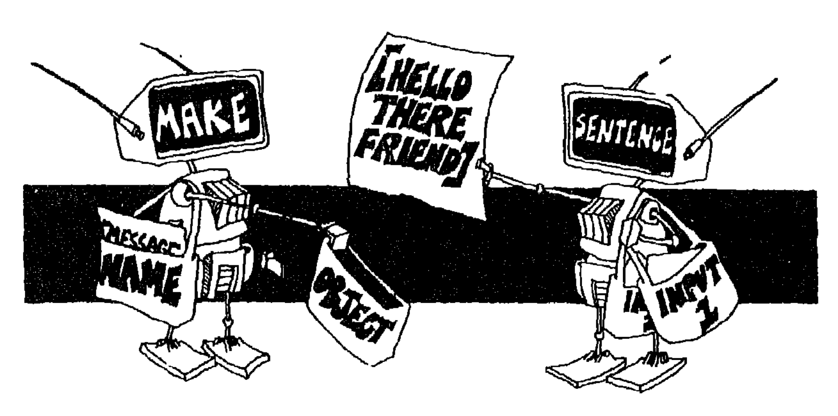
وها هو يلبى أمر الطباعة PRINT فيسلم الروبوط PRINT نسخة من القائمة التى نريد طباعتها .



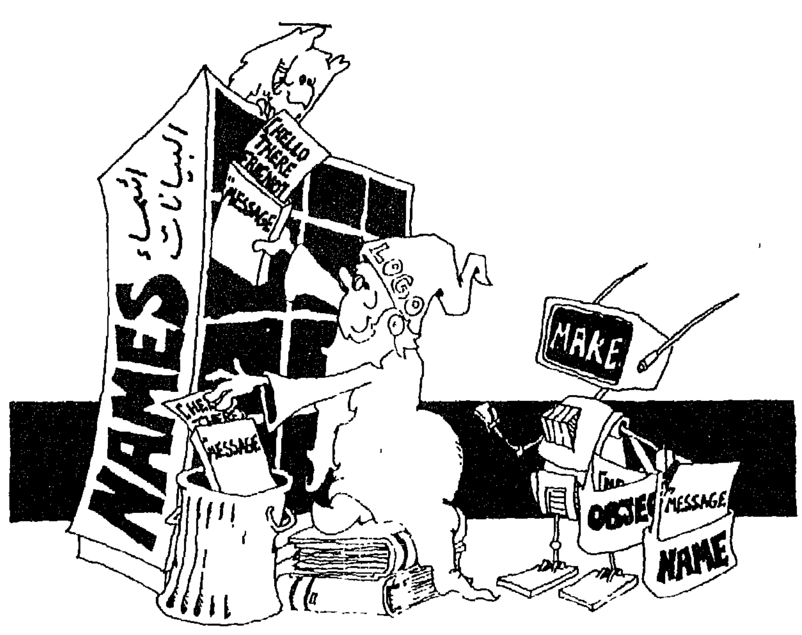
ثم يأتى الروبوط MAKE ومعه عملية جديدة تحتوى على اسم المتغير MESSAGE والبيان المركّب SENTENCE :MESSAGE "FRIEND ولكن البيان لا زال يحتاج إلى مزيد من المعالجة . لذلك يأمر الساحر لوجو الروبوط MAKE باستدعاء الروبوط SENTENCE للقيام بعملية الوصل بين محتوى المتغير MESSAGE وبين الكلمة FRIEND لتكوين الجملة المطلوبة .



يقوم الروبوط MAKE باستدعاء الروبوط SENTENCE ويسلمه بيانات الجملة المطلوب تكوينها فى صورة مُدخلين هما "FRIEND و [HELLO THERE] .

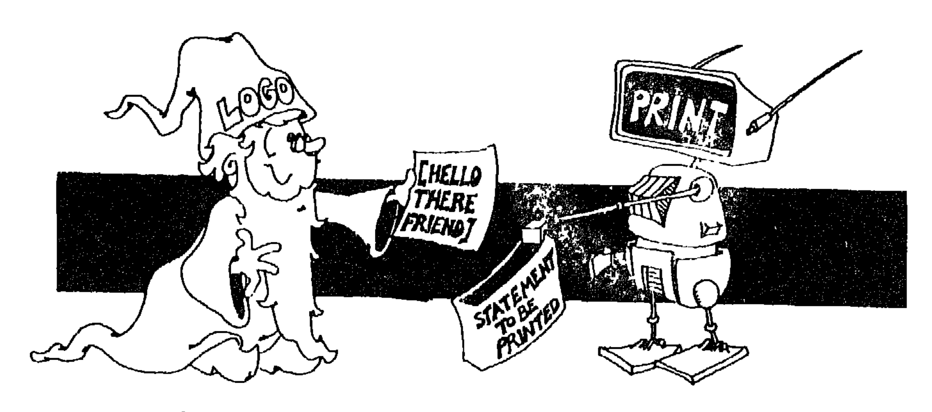


يقوم الروبوط SENTENCE بعملية الوصل المطلوبة ويسلم الروبوط MAKE القائمة [HELLO THERE MY FRIEND] .



وصلت محتويات الجملة SENTENCE الآن .

لذلك لا يحتاج الساحر لوجو للمكونات الأساسية التى كانت موجودة فى علبة المتغيّر MESSAGE فيلقيها فى سلة المهملات .



الآن فقط يستطيع الساحر لوجو أن يستدعى الروبوط PRINT لكى يسلمه العبارة المطلوب طباعتها كاملة :

HELLO THERE FRIEND

● **مثال : برنامج لوصل الكلمات والقوائم :**

```
TO GROW :MESSAGE
PRINT [TYPE SOMETHING NEW]
MAKE "NEWPART READLIST
MAKE "MESSAGE SENTENCE :MESSAGE :NEWPART
PRINT [THE MESSAGE IS NOW]
PRINT :MESSAGE
GROW :MESSAGE
END
```

يقوم هذا البرنامج باستقبال ما تدخله إليه من كلمات أو قوائم ووصلها ببعضها البعض بلا توقف وهذا مثال لاستخدام البرنامج الذى يبدأ بالأمر .

? GROW [HOW]

فيطبع على الشاشة الرسالة (أكتب شيئاً جديداً ..) :

TYPE SOMETHING NEW

وكل ما تكتبه يضمه إلى الكلمة HOW ويطبعه على الشاشة ثم يسألك من جديد أن تكتب شيئاً جديداً دون توقف .

وفى المثال الموضح نستخدم البرنامج فى تكوين العبارة :

HOW ARE YOU GETTING ON

ثم نوقف البرنامج من الخارج .

```
?
TYPE SOMTHING NEW

?
?GROW [HOW]
TYPE SOMTHING NEW
 ARE
THE MESSAGE IS NOW
HOW ARE
TYPE SOMTHING NEW
 YOU
THE MESSAGE IS NOW
HOW ARE YOU
TYPE SOMTHING NEW
 GETTING ON
THE MESSAGE IS NOW
HOW ARE YOU GETTING ON
TYPE SOMTHING NEW

STOPPED!!! in GROW
```

إيقاف البرنامج

● استخدام القوائم فى ألعاب الكلمات :

العملية الشرطية يمكن أن تستخدم لتنظيم الكثير من العمليات الجارية سواء على الأرقام أو على الكلمات والقوائم .

وأحد أنواع الألعاب الكومبيوترية هى ألعاب الكلمات أو ألعاب المعلومات بدقة أكبر . فالامتحانات بأنواعها تندرج تحت ألعاب المعلومات . ولنر معاً هذا المثال الذى يختبرنا فيما حصلناه من لغة لوجو :

● مثال (١) امتحان لوجوا :

```
   TO QUIZ1
1  PRINT[WHAT IS THE SHORT FORM OF
   FORWARD]
2  MAKE "ANSWER1 READLIST
3  IF :ANSWER=[FD] [PRINT [YOU GOT IT]
```

```
   STOP ]
4  PRINT [NO, NOT CORRECT PLEASE TRY
   AGAIN...]
5  QUIZ1
   END
```

وقد رقّمنا سطور البرنامج للإيضاح فقط (فيما عدا اسم البرنامج وأمر النهاية) .

١ ــ ففى السطر الأول يطبع البرنامج على الشاشة العبارة التى تحتويها القائمة وهى :

WHAT IS THE SHORT FORM OF FORWARD

بمعنى ما هو اختصار الأمر FORWARD ؟

٢ ــ ويتلقى الكومبيوتر الإجابة فى السطر الثانى بموجب الأمر READLIST ويخصصها للمتغير ANSWER1 بعملية التخصيص MAKE .

٣ ــ يقوم البرنامج فى السطر الثالث باختبار محتويات المتغير ANSWER1 بواسطة العملية الشرطية .

فإذا كانت محتويات المتغير ANSWER1 مساوية للقائمة FD كانت الإجابة صحيحة وقام الكومبيوتر بتنفيذ القائمة المركبة التالية وهى تتكون من جزئين أولهما طبع الرسالة « YOU GOT IT » والجزء الثانى هو الأمر STOP الذى يجعله يتوقف .

٤ ــ أما إذا لم تكن الإجابة هى FD فسوف ينفذ السطر الرابع الذى يحتوى على الرسالة « الإجابة غير صحيحة حاول مرة أخرى .. » .

٥ ــ وبذلك يتم الانتقال إلى أول البرنامج بموجب الأمر QUIZ1 فى السطر الخامس الذى هو عبارة عن اسم البرنامج نفسه .

● **مثال (٢) امتحان لوجو C :**

أما هذا البرنامج فهو صورة مطوّرة للبرنامج السابق حيث يمنحك اختياراً أن تحاول مرة أخرى أو لا تحاول .

```
   TO QUIZ2
1  PRINT [WHAT IS THE SHORT FORM OF FORWARD?]
2  MAKE "ANSWER2 READLIST
3  IF :ANSWER2 = [FD] [PRINT [CORRECT!] STOP]
4  PRINT [NO, THAT'S NOT IT.]
5  PRINT [WOULD YOU LIKE TO TRY AGAIN?]
6  MAKE "TRY READLIST
7  IF :TRY = [NO] [PRINT [THE ANSWER IS: FD] STOP]
8  IF :TRY = [YES] [QUIZ2 STOP]
9  PRINT [I QUIT! YOU DIDN'T ANSWER YES OR NO.]
   END
```

عند تشغيل هذا البرنامج فإنه يطبع الرسالة التى تحتوى عليها القائمة فى السطر الأول :

WHAT IS THE SHORT FORM OF FORWARD?

بمعنى ما هى الصيغة المختصرة للأمر FORWARD ؟

فى السطر الثانى يتلقى منك الكومبيوتر الإجابة بالأمر **READLIST** ويخصصها للمتغير ANSWER2 .

بعد ذلك يتم اختيار محتوى المتغير ANSWER2 فى السطر الثالث الذى يحتوى على أوامر العملية الشرطية . فإذا كانت الإجابة هى **FD** فإن الكومبيوتر يطبع الرسالة CORRECT ويتوقف بموجب الأمر STOP . وإلاّ ..

وإلاّ فإنه يطبع الرسالة الواردة فى السطر الرابع :

NO THAT'S NOT IT

ويلى ذلك طبع السؤال الذى فى السطر الخامس « هل تحاول مرة أخرى ؟ » .

WOULD YOU LIKE TO TRY AGAIN ?

ويستقبل إجابة جديدة فى السطر السادس ويضعها فى المتغير TRY"

وفى السطر السابع يختبر محتوى المتغير TRY بالعملية الشرطية . فإذا كان يحتوى على القائمة [NO] فإن الكومبيوتر يطبع الإجابة الصحيحة ثم يتوقف .

وفى السطر الثامن يختبر ما إذا كانت الإجابة هى [YES] . فإذا كانت كذلك يعود إلى أول البرنامج لتنفيذ الأمر الأول فى القائمة وهو :

QUIZ2

وهذا هو اسم اللعبة نفسها .

أما السطر التاسع فيتم تنفيذه فى حالة ما إذا كانت الإجابة شيئاً مختلفاً غير (YES) أو (NO) فهذان هما الإجابتان اللتين يقبلهما الكومبيوتر أما ما عدا ذلك فإنه يغضب ويرسل الرسالة التى يحتوى عليها السطر التاسع وهى تقول « سوف أرحل لأنك لم تجبنى بنعم أولاً . وينتهى البرنامج عند هذا الحد .

● العمليات المنطقية OR, AND

إذا أردت السفر من القاهرة إلى الإسكندرية فيمكنك استخدام القطار أو « الأوتوبيس » أينما تجد مقعداً خالياً .

فإذا أردت إرسال إنسان آلى (روبوط) ليقوم بشراء تذكرة سفر لك فعليك أن تشرح له هذه المسألة بلغة يفهمها .

وفى لغة لوجو نستخدم الأمر OR الذى يدخل مع الأمر IF فى عملية مركّبة للتعبير عن ذلك :

احجز	مقعد فى الأتوبيس	مقعد فى القطار		
التصرف فى أى من الحالين	الشرط الثانى	الشرط الأول	OR	IF

والشرط الأول هنا هو وجود مقعد خال بالقطار .

والشرط الثانى هو وجود مقعد خال بالأتوبيس .

والتصرف المطلوب هو الحجز .

فلو تحقق أحد الشرطين أو كلاهما فسوف يتم الحجز والسفر .

وتسمى هذه العملية بعملية الجمع المنطقى لأنها . من وجهة نظر الكومبيوتر عملية منطقية لها أحد نتيجتين إما صحيح TRUE أو خطأ FALSE فهو يختبر الشرط الأول فإذا وجده صحيحاً أى كانت نتيجته TRUE اكتفى به واعتبر أن العملية كلها صحيحة ولذلك فهو ينفذ التصرف المطلوب .

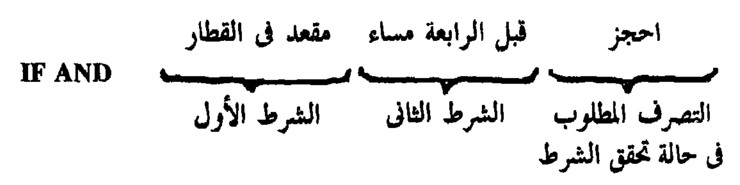
فإذا لم يتحقق الشرط الأول (أى كانت نتيجته FALSE) اختبر الشرط الثانى أيضاً فإذا كان صحيحاً قام أيضاً بتنفيذ التصرف المطلوب . حالة واحدة هى التى تفشل فيها العملية OR وهى عندما لا يجد الروبوط مقعداً خالياً فى القطار أو فى الأتوبيس . بذلك يعطى كل من الشرطين النتيجة FALSE ولاتتحقق العملية الشرطية فينفذ القائمة التالية بعد ذلك ، أياً كان مضمونها .

هناك عملية منطقية أخرى هى العملية AND وهى أكثر تشدّداً من العملية السابقة .

ولنفرض أنك طلبت من الروبوط تحقيق شرط آخر وهو أى موعد القطار قبل الرابعة مساء .

بهذا تخضع عملية حجز تذكرة القطار لشرط مرّكب وهو وجود مقعد خال وأن يكون الموعد قبل الرابعة .

هذا يتم التعبير عنه على الصورة :



وتسمى العملية AND بعملية الضرب المنطقى .

فلنر مثلاً بسيطاً باستخدام الأعداد :

```
IF   OR   1=1   2=5   [ PRINT [ OK, SIR ]   STOP ]
PRINT   [ NOT   OK ]
```

ماذا سيطبع الكومبيوتر فى هذا المثال الرسالة الأولى أم الثانية ؟

إن الشرط الأول 1 = 1 يعطى القيمة TRUE .

أما الشرط الثانى 5 = 2 فيعطى القيمة FALSE .

ونظراً لأن الأمر OR يكتفى بتحقق شرط واحد لذلك فهو سيعتبر أن العملية الشرطية قد نجحت ويعطى الأمر OR القيمة TRUE كنتيجة نهائية فيطبع بذلك الرسالة الأولى وهى :

OK , SIR

ثم يتوقف البرنامج .

ولو استبدلنا الشرط 1 = 1 بشرط آخر مثل 2 = 1 فإن الرسالة الثانية هى التى ستطبع .

ولنر مثالاً للعملية AND باستخدام الأرقام أيضاً .

```
IF   AND   1 = 1   2 = 5   [ PRINT [ OK, SIR ]   STOP ]
PRINT   [NOT OK]
```

فى هذه الحالة لن تتحقق نتيجة العملية الشرطية لأن الأمر AND سوف يُرجع القيمة FALSE لأن أحد الشرطين لم يتحقق .

● مثال (٣) امتحان لوجو ٣ :

وبذلك نصل إلى تطوير جديد فى هذا المثال حيث أن السؤال الواحد فى بعض الأحيان قد تكون له أكثر من إجابة صحيحة . ولنر هذا السؤال :

" كم من المُدخلات (INPUTS) يحتاج إليها الأمر MAKE ؟" والإجابة على هذا السؤال قد تكون بالحروف (TWO) أو بالأرقام (2) وكلاهما إجابة صحيحة .

ولتلقين الكومبيوتر بهذه الاحتمالات يجب استخدام الأمر (OR) .

ولنر المثال :

```
  TO QUIZ3
1 PRINT [HOW MANY INPUTS DOES THE]
2 PRINT [MAKE COMMAND NEED?]
3 MAKE "ANS3 READLIST
4 IF OR :ANS3 = [TWO] :ANS3 = [2] [PRINT [RIGHT.] STOP]
5 PRINT [NO, MAKE NEEDS TWO INPUTS,]
6 PRINT [A NAME AND A VALUE]
  END
```

١ ، ٢ ــ الرسالة التى تظهر على الشاشة فى البداية قد تم طبعها فى السطرين 1 ، 2 وسوف تظهر مكتوبة على سطرين كما هى .

٣ ــ تُخصص الإجابة للمتغير ANS3 .

٤ ــ يتم اختبار الإجابة ANS3 ما إذا كانت مساوية [2] أو [TOW] وفى أى من الحالتين تطبع الرسالة RIGHT ثم يتوقف البرنامج .

٥ ، ٦ ــ فى حالة احتواء الإجابة ANS3 على أى شىء خلاف القيمتين [2] أو [TWO] تُطبع الرسالة :

NO, MAKE NEEDS TWO INPUTS

A NAME AND A VALUE

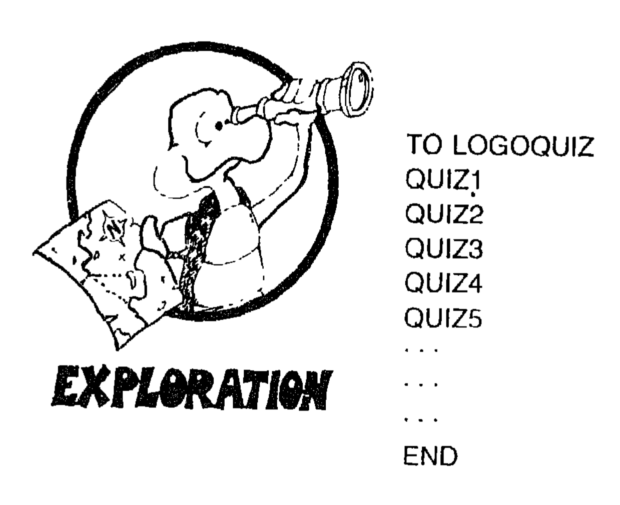
وينتهى البرنامج عند هذا الحد .

مثل هذا البرنامج الذى يضع الاحتمالات المختلفة للإجابات يوصف بأنه برنامج ذكى . أما البرنامج الذى يعترف بالرقم 2 ولا يعترف بكلمة TWO فهو برنامج غبى

وبالطبع لا يتوقف الأمر على مجرد احتمالين فهناك بعض الأسئلة تحتمل إجابات كثيرة أو ربما تحتاج إلى مناقشة وفى هذه الحالة يجب تغذية الكومبيوتر بكمية كبيرة من المعلومات .

ومن خصائص الامتحان الذكى أيضاً تتبع للإجابات جميعاً وتقييمها لإعطاء درجة نهائية أو تقدير نهائى .

وبوضع العديد من البرامج الفرعية التى يحتوى كل منها على سؤال مستقل يمكن بناء امتحان متكامل ببرنامج رئيسى بالصورة الآتية :




```
TO LOGOQUIZ
QUIZ1
QUIZ2
QUIZ3
QUIZ4
QUIZ5
. . .
. . .
. . .
END
```

● تمرير البيانات من برنامج إلى آخر OUTPUT (OP)

لعلنا لم نلتق من قبل بلفظة « التمرير » إلاّ فى مباريات كرة القدم حيث يقوم أحد اللاعبين بتمرير الكرة إلى لاعب آخر ليضعها فى المرمى مسجلاً هدفاً .

ونحن الآن أمام متغير وظيفته تنحصر أساساً فى « تمرير » قيم المتغيرات إلى البرنامج الرئيسى لكى يستخدمها المبرمج فيما يشاء .

ولنر المثال الآتى :

```
TO MEAN :X :Y
OUTPUT (:X + :Y)/2
END
```

هذا البرنامج لا يطبع شيئاً على الشاشة ولكنه يحسب قيمة التعبير الرياضى ويمرر قيمته إلى البرنامج (أو الأمر المباشر) الذى يستدعيه كالمثال الآتى :

الأمـــــــــــــر	←	**PR MEAN 6 3**
الجواب	←	**4.5**

وكما نرى فإن عملية التمرير تتم عندما يُستدعى البرنامج المحتوى على الأمر OUTPUT .

لذلك نرى أن هناك فارقاً بين البرنامج الذى ينتهى بالأمر END والبرنامج الذى ينتهى بالأمر OUTPUT . فالأول يُعامل كأنه أمر من أوامر لوجو ويتم تنفيذه بمجرد كتابة اسمه . أما الأخير فإنه يعتبر عملية مثل عملية الجذر التربيعى وجب الزاوية ولا يستخدم إلاّ من خلال برنامج أو أمر مباشر .

والأمر OUTPUT يعتبر نهاية البرنامج لأنه يوقف التنفيذ تماماً كما يفعل الأمر STOP مع فارق أنه يضع فى الذاكرة قيمة المتغير أو التعبير الذى يتبعه والذى نطلق عليه اسم « الهدف » (object) .

● اختبار الأعداد NUMBERP

أحياناً نحتاج لمعرفة ما إذا كان أحد المتغيرات يحتوى على قيمة عددية أم على قائمة . فمثلاً عندما نقرأ عدداً ما بالأمر READLIST فإنه يُختزن فى صورة قائمة ولا يمكن إجراء أية عمليات حسابية عليه فى هذه الحالة .

ويمكن إجراء هذا الأختبار بالأمر NUMBERP كالآتى :

● مثال [١] :

الأمر	←	**PR NUMBERP 3**
الجواب	←	**TRUE**

● مثال [٢] : الأمر ← PR NUMBERP [3]

الجواب ← FALSE

● مثال [٣] :

```
TO TEST1
MAKE "N 55
PR NUMBERP N
END
```

عند تشغيل هذا البرنامج يعطى القيمة TRUE

● مثال [٤] :

```
TO TEST2
MAKE "N RL
PR NUMBERP N
END
```

وعند تشغيل هذا البرنامج فإنه يعطى القيمة FALSE

● **النفى المنطقى NOT :**

يستخدم النفى المنطقى لعكس النتيجة كالأمثلة الآتية :

الأمر ← PRINT 4 = 5

الجواب ← FALSE

الأمر ← PRINT NOT 4 = 5

الجواب ← TRUE

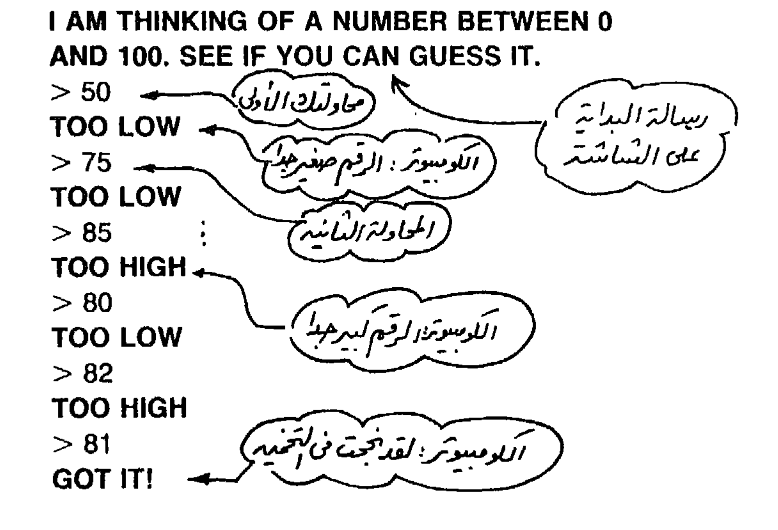
الأمر ← PRINT NOT NUMBERP 3

الجواب ← FALSE

المثال الأخير معناه « هل البيان 3 لا يُعتبر عدداً ؟ » والإجابة بالطبع هى « بلى » .

فى عقل الكومبيوتر رقم معيّن يعرفه لكننا لا نعرفه ومطلوب منا أن نخمّن هذا الرقم والكومبيوتر يتلقى إجابتنا مرة بعد مرة ويحاول جاهداً أن يقربنا من الإجابة الصحيحة .

لنر أولاً كيف يعمل هذا البرنامج على الشاشة :



يبدأ الكومبيوتر بطبع الرسالة الموضحة على الشاشة يقول فيها :

« إننى أفكر فى رقم بين الصفر والمائة ... هل تستطيع تخمينه ؟ » . ثم تحاول أنت أن تكتب رقماً جزافياً مثل 50 .

فيرد عليك الكومبيوتر قائلاً : « هذا الرقم أصغر بكثير » .

وهذا يجعلك فى المحاولة التالية تكتب رقماً أكبر .. 75 مثلاً .

فيقول .. « هذا الرقم أصغر بكثير » .

إذن نحاول أن نكتب رقماً أعلى من ذلك .. 85 مثلاً .

فيرد علينا قائلاً « هذا الرقم كبير جداً » .

وهكذا حتى تتحقق المحاولة الناجحة .. وفى هذا المثال تكون 81 .

● فيكف ننشىء هذا البرنامج ؟ إن العمل سوف يتكون من أجزاء أربعة هى :

١ ـــ كتابة تعليمات اللعبة على الشاشة .

٢ ـــ اختيار رقم ما بين 0 ، 100 .

٣ ـــ استقبال تخمينات مستخدم البرنامج .

٤ ـــ إجراء اختبار على كل رقم يدخله المستخدم وطبع رسالة تخبره ما إذا كان الرقم أكبر أو أصغر من الرقم المقصود . وعلى ذلك يمكننا بناء البرنامج كالآتى :




```
TO GUESSNUMBER
INSTRUCTIONS
CHOOSENUMBER
GETGUESS
END
```

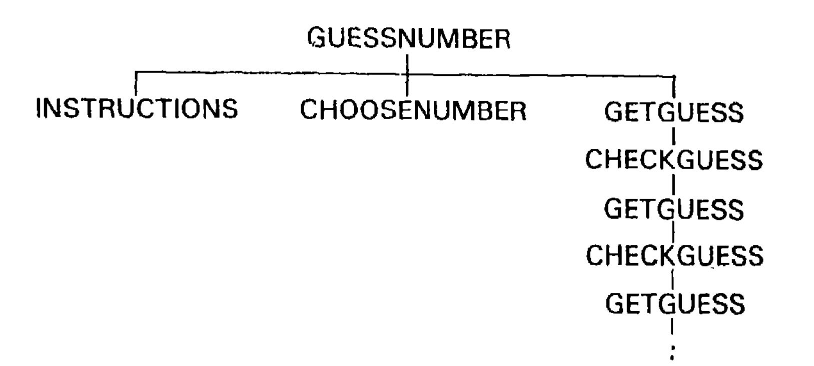



```
1 TO INSTRUCTIONS
CLEARTEXT
PRINT [I AM THINKING OF  A NUMBER BETWEEN 0]
PRINT [AND 100. SEE IF YOU CAN GUESS IT.]
END
2 TO CHOOSENUMBER
MAKE "NUMBER 1 + RANDOM 99
END
3 TO GETGUESS
TYPE ">
MAKE "GUESS READNUMBER
CHECKGUESS :GUESS :NUMBER
END
4 TO CHECKGUESS :GUESS :NUMBER
IF :GUESS = :NUMBER [PRINT [GOT IT] STOP]
IF :GUESS > :NUMBER [PRINT [TOO HIGH] GETGUESS STOP]
IF :GUESS < :NUMBER [PRINT [TOO LOW] GETGUESS STOP]
END
```

البرنامج الرئيسى هو GUESSNUMBER

وهو يستدعى ٣ برامج فرعية : INSTRUCTIONS للتعليمات . CHOOSENUMBER لاختيار الرقم GETGUESS لاختبار الرقم المُدخل .

والبرنامج الأخير يستدعى برنامجاً فرعياً آخر يحتوى على العمليات الشرطية المختلفة ويتناوب البرنامجان الأخيران استدعاء بعضهما البعض عدة مرات وحتى الوصول للرقم المطلوب . أى أنه يمكن تصور الشجرة الآتية لتنفيذ البرامج :



● ولنناقش الآن البرامج الفرعية .

تظهر لنا مبدئياً عند القراءة الأولى لهذه البرامج الفرعية عبارات جديدة من لغة لوجو . هى : READNUMBER , CLEARTEXT , TYPE .

*** الأمر TYPE**

هذا الأمر يماثل تماماً الأمر PRINT ولكنه لا يجعل النقطة الضوئية (cursor) تنتقل إلى السطر التالى بمعنى أن الطباعة المتتالية بالأمر TYPE تكون على نفس السطر .

*** الأمر CLEARTEXT واختصاره CT**

هذا الأمر يستخدم لتنظيف الشاشة من الكتابة عندما تكون الشاشة فى طور الكتابة (TEXTSCREEN) .

* الأمر READNUMBER

يكافىء الأمر READLIST ولكنه مخصص لقراءة الأرقام فهو يجعل الكومبيوتر ينتظر إدخال رقم من الأرقام علاوة على أنه يقوم باختبار ما إذا كان المدخل هو رقم فعلاً أم قائمة أو حتى قائمة خالية [ ] .

وهذا الأمر ليس من الأوامر القياسية للغة لوجو . بل هو برنامج فرعى ضمن البرامج التى يحتوى عليها القرص (LWIL) . وقد عرضنا نص هذا البرنامج فى الملحق رقم (٣) ويمكنك نسخه واستخدامه كما هو .

ولكنه قد يتصادف أيضاً أن يحتوى البرنامج الفرعى READNUMBER على أوامر لا تتوفر فى طراز اللغة الذى يستخدمه كومبيوتر معين مثل الأمر TEST وفى هذه الحالة يفضل بناء البرنامج بنفسك باستخدام الأوامر المتاحة بطراز اللغة الذى تعمل عليه .

والبرنامج الآتى بعد يحمل نفس الاسم READNUMBER أيضاً ولكنه يستخدم العملية الشرطية IF بدلاً من الأمر TEST حتى يصلح لأجهزة الكومبيوتر الصغيرة مثل سنكلير . ولا بأس من استخدام هذا البرنامج الفرعى أيضاً .

```
TO READNUMBER
MAKE "NUM1 RL
IF :NUM1 = [] [PR [PLEASE TYPE A
 NUMBER] OUTPUT READNUMBER]
IF NOT NUMBERP FIRST :NUM1 [PR [
PLEASE TYPE A NUMBER] OUTPUT REA
DNUMBER]
OUTPUT FIRST :NUM1
END
```



هذه هى الأوامر الجديدة التى نراها فى مجموعة البرامج الفرعية للعبة للرقم السرى .

أما اللعبة نفسها فتسير أحداثها كالآتى :

اللعبة تتكون أساساً من برنامج رئيسى هو GUESSNUMBER الذى يستخدم ٣ برامج فرعية كالآتى :

```
TO GUESSNUMBER
INSTRUCTIONS
CHOOSENUMBER
GETGUESS
END
```

● البرنامج الفرعى الأول هو برنامج التعليمات INSTRUCTIONS الذى يقوم بطباعة تعليمات اللعبة على الشاشة :

```
TO INSTRUCTIONS
CLEARTEXT
PRINT [I AM THINKING OF A NUMBER BETWEEN 0]
PRINT [AND 100. SEE IF YOU CAN GUESS IT.]
END
```

فى هذا البرنامج نستهلّ بمسح الشاشة فى طور الكتابة بالأمر CLEARTEXT ثم نطبع تعليمات اللعبة على سطرين متعاقبين .

● يلى ذلك البرنامج الفرعى لاختيار الرقم السرّى CHOOSENUMBER

```
TO CHOOSENUMBER
MAKE "NUMBER 1 + RANDOM 99
END
```

وفى هذا البرنامج نستخدم الأمر RANDOM لتوليد عدد يتراوح بين الصفر و 98.ثم نضيف إليه الرقم 1 فتصبح قيمته عندئذ بين الصفر و 99 . ونخصص هذا العدد العشوائى للمتغير NUMBER بواسطة أمر التخصيص MAKE .

● البرنامج الفرعى التالى هو GETGUESS الذى يستقبل الرقم المُدخل من مستخدم البرنامج ويجرى عليه الاختبارات اللازمة :

```
TO GETGUESS
1 TYPE ">
2 MAKE "GUESS READNUMBER
3 CHECKGUESS :GUESS :NUMBER
END
```

يبدأ البرنامج بطبع العلامة (<) :الأمر TYPE وذلك حتى لا تنتقل النقطة الضوئية إلى السطر التالى . وبذلك فإن أى كتابة تالية سوف تكون على نفسه السطر .

أما السطر التالى من البرنامج فهو يخصص الرقم المُدخل للمتغيّر GUESS . وفى السطر الثالث يتم استدعاء البرنامج الفرعى GHECKGUESS الذى يستخدم دليلين أحدهما هو المتغيز GUESS الذى يحتوى على الرقم المُدخل بواسطة المستخدم والثانى هو MUMBER وهو الرقم السرى الذى يخبئه الكومبيوتر . ومن المتوقعّ أن تتم المقارنة بين الرقمين بواسطة البرنامج الفرعى الأخير .

● البرنامج الفرعى **CHECKGUESS** لاختبار صحة التخمين :

```
TO CHECKGUESS :GUESS :NUMBER
1 IF :GUESS = :NUMBER [PRINT [GOT IT] STOP]
2 IF :GUESS > :NUMBER [PRINT [TOO HIGH] GETGUESS STOP]
3 IF :GUESS < :NUMBER [PRINT [TOO LOW] GETGUESS STOP]
END
```

كما نرى أن هذا البرنامج هو برنامج لدالة تستخدم دليلين GUESS ، NUMBER وتقارن بينهما باستخدام العملية الشرطية .

وفى السطر الأول يتم اختبار التساوى بين الرقم المُدخل والرقم السرى فإذا صحّ الاختبار فإن الرسالة "GOT IT" تظهر على الشاشة وينتهى الأمر .

أما إذا كان الرقم أكبر كما فى السطر الثانى فإن الرسالة "TOO HIGH" هى التى تطبع ويتم التفرع إلى البرنامج GETGUESS لاستقبال رقم جديد .

وأما إذا كان الرقم أصغر فتطبع الرسالة "TOO LOW" ويتم التفرع أيضاً

إلى البرنامج GETGUESS لاستقبال رقم جديد .

وعند العودة إلى البرنامج GETGUESS يستقبل البرنامج تخميناً جديداً ثم يعود لاختباره بواسطة البرنامج CHECKGUESS وهكذا حتى نصل إلى الرقم الذى فى عقل الكومبيوتر .

والآتى بعد هو طبعة لكل ما فى حيز العمل أثناء تشغيل هذه البرامج ونلاحظ ضرورة وجود البرنامج READNUMBER فى حيز العمل . كما نرى مجموعة المتغيرات العامة التى استخدمت فى البرامج علاوة على بعض متغيرات أخرى مثل L ، S وهى تابعة لبرامج أخرى تم مسحها من بالأمر (ER) .

```
?
TO GUESSNUMBER
TO CHECKGUESS :GUESS :NUMBER
TO INSTRUCTIONS
TO GETGUESS
TO CHOOSENUMBER
TO READNUMBER

?
TO GUESSNUMBER
INSTRUCTIONS
CHOOSENUMBER
GETGUESS
END

TO DHECKGUESS :GUESS :NUMBER
IF :GUESS = :NUMBER [PRINT [GOT
IT] STOP]
IF :GUESS > :NUMBER [PRINT [TOO
HIGH] GETGUESS STOP]
IF :GUESS < :NUMBER [PRINT [TOO
LOW] GETGUESS STOP]
END

TO INSTRUCTIONS
CT
PR [I AM THINKING OF A NUMBER BE
TWEEN 0]
PR [AND 100 . SEE IF YOU CAN GUE
SS IT]
END

TO GETGUESS
TYPE ">
MAKE "GUESS READNUMBER
CHECKGUESS :GUESS :NUMBER
END

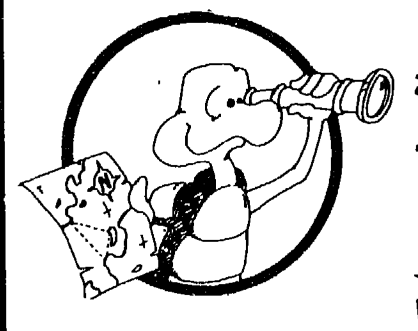
TO CHOOSENUMBER
MAKE "NUMBER 1 + RANDOM 99
END

TO READNUMBER
MAKE "NUM1 RL
IF :NUM1 = [] [PR [PLEASE TYPE A
 NUMBER] OUTPUT READNUMBER]
IF NOT NUMBERP FIRST :NUM1 [PR [
PLEASE TYPE A NUMBER] OUTPUT REA
DNUMBER]
OUTPUT FIRST :NUM1
END

MAKE "GUESS "70
MAKE "L "555
MAKE "S "14.323957
MAKE "NUMBER "6
MAKE "NUM1 [6]
```

متغيرات لعبة الرقم السري



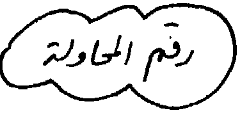


هل تستطيع أن تجعل اللعبة محتوية على عدّاد يقوم بحساب عدد المحاولات ؟

إن هذا يتطلب استخدام متغير جديد تزيد قيمته بمقدار 1 كلما كلما أدخلنا رقماً جديداً .

وبعد إضافة مثل هذا التعديل إلى البرنامج فإن أحد الصور المقبولة له تكون كالآتى حيث يكتب رقم المحاولة أمام العلامة" > " باستمرار :

```
GUESSNUMBER2
I AM THINKING OF A NUMBER BETWEEN 0
AND 100. SEE IF YOU CAN GUESS IT.
1 > 50
TOO HIGH
2 > 25
TOO LOW
3 > 35
TOO HIGH
4 > 30
TOO HIGH
5 > 28
GOT IT!
```



حاول .. واستمتع بالنتائج .

● لعبة تعليمية امتحان فى الرياضة MATHQUIZ

من أجمل الألعاب التى يمكن أن نستمتع بها مع الكومبيوتر هى الألعاب التعليمية لأنها دائماً تأتى بالجديد ، فهى لا تفقد عنصر الإثارة أبداً . بينما مع ألعاب القذائف والحروب عادة ما تبدأ اللعبة مثيرة للحماس والرغبة فى تحطيم الرقم المسجّل من أحد الأصدقاء المهرة لكنك بمجرد أن تتدرب عليها وتستطيع تسجيل رقم قياسى تبدأ فى البحث عن غيرها إذ تفقد الإثارة والتشويق .

والألعاب التعليمية تصل إلى ذروتها فى الإبداع والإثارة عندما تجمع بين الهدف التعليمى وبين الإبداع الفنى فى تكوين « قصة » اللعبة وتدعيمها بالرسم الجميل والألوان والمؤثرات الصوتية .

ولا نتوقع بالطبع أن نقدم فى كتاب تعليم لغة لوجو كل المهارات التى تجعلنا نصمم لعبة كومبيوترية كاملة بلغة لوجو . فنحن هنا نقدم المنطق الأساسى للعبة أما الرسم والتلوين والموسيقى والتحويرات المختلفة التى تجعل منها لعبة منافسة للألعاب الكومبيوترية الأخرى فكل هذا يحتاج منا المزيد من المهارة والتدريب على بناء الألعاب الكومبيوترية . ولعلنا فى لقاء قريب إن شاء الله نلتقى حول برمجة الألعاب بلغة لوجو كما التقينا من قبل حول برمجة الألعاب بلغة بيسك .

واللعبة التى نحن بصددها الآن عبارة عن أسئلة فى الرياضة يطرحها عليك الكومبيوتر ويصحح لك الإجابات ويمنحك درجة فى النهاية .

هذا هو أساس اللعبة .

أما الإضافات التى يمكن أن نضيفها إليها فلا حصر لها .

فبدلاً من الأرقام المجردة يمكن استخدام مركبات الفضاء التى تغزو الأرض من الفضاء الخارجى والتى لا يمكن إصابتها إلاّ بضربها بمدافع « الأجوبة الصحيحة » !

وبصفة عامة فأياً كان شكل اللعبة فهى تعتمد على منطق أساسى فى البناء . وهذا هو الذى نقدمه لك الآن .

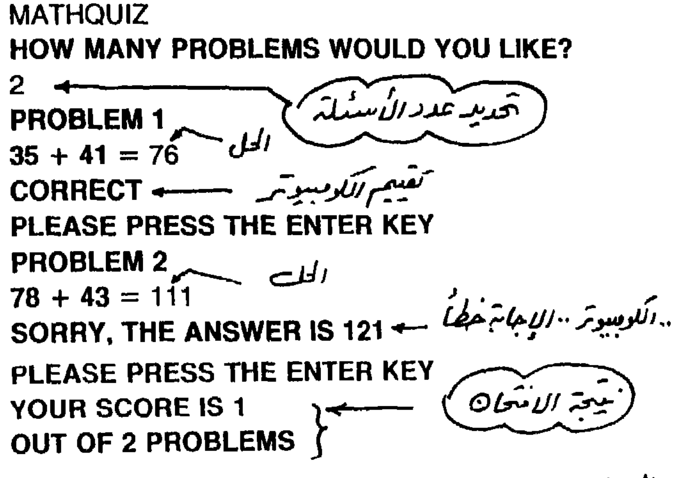
● تشغيل اللعبة :

عندما تكتب اسم اللعبة على الشاشة فإنها تُمسح وترىَ على الشاشة السؤال الآتى :

HOW MANY PROBLEMS WOULD YOU LIKE ?

وذلك أن تحدد عدد الأسئلة التى يتضمنها الامتحان فيبدأ بعدها الكومبيوتر فى تقديم واحداً تلو الآخر .

والآتى بعد مثال للحوار الدائر على الشاشة :



● بناء البرنامج :

بهذه الصورة السابقة للبرنامج يمكن أن نتصور أنه مبنى بالأسلوب الآتى :

١ ــ اختيار عدد المسائل المطلوب حلّها .

٢ ــ اختيار عددين تمثيل المسألة الأولى .

٣ ـــ اختبار الإجابة وطبع نتيجة الاختبار CORRECT أو SORRY مع زيادة عدّاد الإجابات الصحيحة بمقدار 1 فى حالة الأولى .

٤ ـــ اختبار بلوغ عدد الأسئلة إلى الحد المطلوب ، فإذا كان كذلك يتم طبع النتيجة النهائية وينتهى البرنامج . وإلاّ فيتم زيادة العدّاد بمقدار 1 واختيار رقمين جديدين .

● البرامـــج :

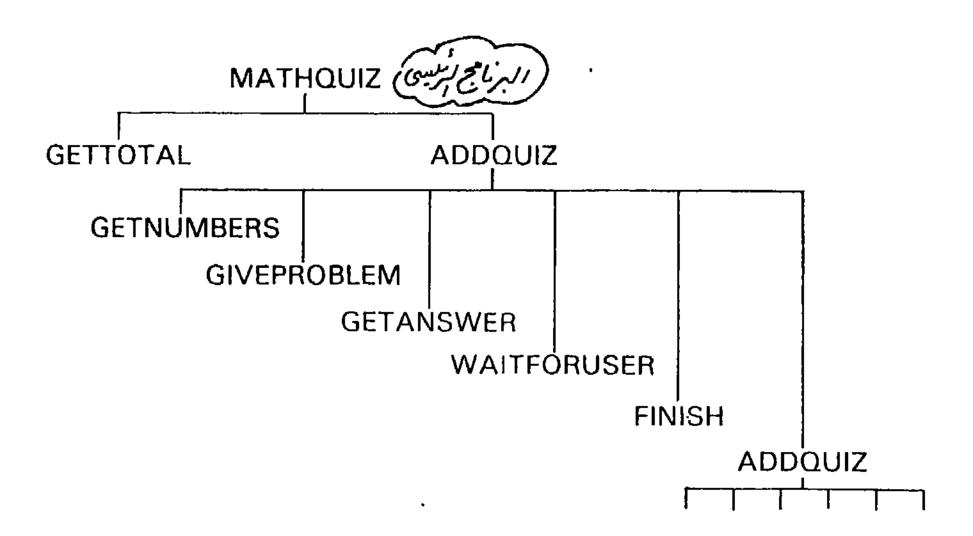
يمكن للأصدقاء المتعجّلين بلوغ النتائج كتابة مجموعة البرامج الآتية والبدء فى تجربة اللعبة فوراً :

```
TO MATHQUIZ  ←  البرنامج الرئيسى
CLEARTEXT
GETTOTAL
MAKE "COUNT1
MAKE "SCORE 0
ADDQUIZ :COUNT :TOTAL :SCORE
END
1 TO GETTOTAL  ←  البرامج الفرعية
PRINT [HOW MANY PROBLEMS DO YOU WANT?]
MAKE "TOTAL READNUMBER
END
2 TO ADDQUIZ :COUNT :TOTAL :SCORE
CLEARTEXT
GETNUMBERS
GIVEPROBLEM
GETANSWER
WAITFORUSER
IF :COUNT = :TOTAL [FINISH STOP]
ADDQUIZ ( :COUNT + 1 ) :TOTAL :SCORE
END
3 TO GETNUMBERS
MAKE "NUMBER1 RANDOM 100
MAKE "NUMBER2 RANDOM 100
MAKE "RIGHTANSWER :NUMBER1 + :NUMBER2
END
4 TO GIVEPROBLEM
PRINT SENTENCE [PROBLEM] :COUNT
PRINT [ ]
TYPE ( SENTENCE :NUMBER1 [+] :NUMBER2 [=] )
END
*5 TO GETANSWER
MAKE "RESPONSE READNUMBER
TEST :RESPONSE = :RIGHTANSWER
IFTRUE [PRINT [CORRECT]]
IFTRUE [MAKE "SCORE :SCORE + 1]
IFFALSE [PRINT SENTENCE [SORRY, THE ANSWER IS]
   :RIGHTANSWER]
END
6 TO WAITFORUSER
PRINT [PLEASE PRESS THE ENTER KEY]
PRINT READLIST
END
7 TO FINISH
CLEARTEXT
PRINT SENTENCE [YOUR SCORE IS] :SCORE
PRINT ( SENTENCE [OUT OF] :TOTAL [PROBLEMS] )
END
```

● **ملاحظة :** البرنامج رقم 5 قد لا يعمل على بعض أجهزة الكومبيوتر ولذلك سوف نستبدله ببرنامج فرعى آخر أثناء المناقشة التالية :

● **مناقشة اللعبة :**

يمكن تصور اللعبة كشجرة من البرامج الفرعية تخرج جميعاً من البرنامج الرئيسى MATHQUIZ وتتسلسل كما فى الشكل التالى :



* **ولنبدأ بالبرنامج الرئيسى MATHQUIZ :**

يبدأ البرنامج بتنظيف الشاشة ثم يستدعى البرنامج الفرعى GETTOTAL لكى يسألك عن عدد الأسئلة المطلوبة فى الامتحان .
وبإدخال عدد الأسئلة يخزنها البرنامج الفرعى GETTOTAL فى المتغير TOTAL .
ثم يقوم البرنامج الرئيسى بوضع قيمة ابتدائية (0) لعدّاد النتيجة SCORE ، وقيمة ابتدائية (1) لعدّاد الأسئلة COUNT . وبهذه المتغيرات الثلاثة يتم استدعاء البرنامج الفرعى ADDQUIZ .

```
TO MATHQUIZ
CLEARTEXT
GETTOTAL
MAKE "COUNT 1
MAKE "SCORE 0
ADDQUIZ :COUNT :TOTAL :SCORE
END
TO GETTOTAL
PRINT [HOW MANY PROBLEMS DO YOU WANT?]
MAKE "TOTAL READNUMBER
END
```

* أما البرنامج ADDQUIZ فهو يستخدم المتغيرات الثلاثة المذكورة كأدلة له .

وهو يبدأ بمسح الشاشة ثم يستدعى البرامج الأربعة الآتية واحداً تلو الآخر :

١ ـ GETNUMBERS لاختيار الأعداد للمسائل وتحديد الإجابة الصحيحة .

٢ ـ GIVEPROBLEM لطبع المسألة على الشاشة .

٣ ـ GETANSWER لاستقبال الإجابة من مستخدم البرنامج واختبارها .

٤ ـ WAITFORUSER لإتاحة بعض الوقت للاعب قبل أن ينتقل إلى المسألة التالية .

بعد استدعاء هذه البرامج واحداً تلو الآخر فإن البرنامج ADDQUIZ يقوم باختبار قيمة المتغيّر COUNT لمعرفة ما إذا كان عدد الأسئلة قد بلغ غايته .

```
TO ADDQUIZ :COUNT :TOTAL :SCORE
CLEARTEXT
GETNUMBERS
GIVEPROBLEM
GETANSWER
WAITFORUSER
IF :COUNT = :TOTAL [FINISH STOP]
ADDQUIZ ( :COUNT + 1 ) :TOTAL :SCORE
END
```

فإذا كان العدّاد COUNT يحتوى على قيمة مماثلة لقيمة المتغير TOTAL يتفرع البرنامج إلى البرنامج الأخير FINISH أو يضاف واحد إلى العدّاد COUNT ويستدعى البرنامج ADDQUIZ نفسه من جديد لحل مسألة جديدة .

* وهذا هو البرنامج الفرعى GETNUMBERS :

```
TO GETNUMBERS
MAKE "NUMBER1 RANDOM 100
MAKE "NUMBER2 RANDOM 100
MAKE "RIGHTANSWER :NUMBER1 + :NUMBER2
END
```

فهو يبدأ بتخصيص عددين عشوائيين NUMBER1 , NUMBER2 ومنهما يستخرج الإجابة الصحيحة لعملية الجمع RIGHTANSWER .

* أما البرنامج GIVEPROBLEM فهو يتسلسل كالآتى :

```
TO GIVEPROBLEM
PRINT SENTENCE [PROBLEM] :COUNT
PRINT [ ]
TYPE (SENTENCE :NUMBER1 [+] :NUMBER2 [=])
END
```

طباعة المسألة على الشاشة

يطبع على الشاشة الجملة المكونة من الكلمة PROBLEM يتبعها رقم المسألة COUNT .

يلى ذلك طباعة سطر خال بالأمر [ ] PRINT .

ثم يقوم البرنامج بطباعة المسألة نفسها وهى عبارة عن مجموع العددين : NUMBER1 ، NUMBER2 .

* وهذا هو البرنامج الفرعى GETANSWER الذى يستقبل إجاباتك :





استقبال واختبار الإجابة

```
*   TO GETANSWER
1   MAKE "RESPONSE READNUMBER
2   TEST :RESPONSE = :RIGHTANSWER
3   IFTRUE [PRINT [CORRECT]]
4   IFTRUE [MAKE "SCORE :SCORE + I]
5   IFFALSE [PRINT SENTENCE [SORRY, THE ANSWER IS]
      :RIGHTANSWER]
    END
```

يستقبل هذا البرنامج إجابتك مستخدماً البرنامج الفرعى الخاص READNUMBER الذى يجب تحميله من القرص المغنطيسى (LWIL) أو كتابته بالطريقة التى قدمناها فى اللعبة السابقة ضمن البرامج الفرعية . وبعد استقبال الإجابة يتم تخصيصها للمتغير . RESPONSE كما فى السطر 1 .

ويقوم البرنامج باختبار وصحة الإجابة باستخدام **الأمر الجديد TEST** الذى يستخدم جنباً إلى جنب مع كل من الأمرين **IFTRUE** ، **IFFALSE** لتكوين عملية شرطية متكاملة .

ويستخدم الأمر TEST متبوعاً بالشرط المطلوب اختبار صحته وهو هنا شرط تساوى الإجابة المدخلة وهى RESPONSE بالإجابة الصحيحة وهى RIGHTANSWER (كما فى السطر 2) .

ثم يأتى السطر الثالث ليقول « إذا كان الاختبار صحيحاً افعل كذا وكذا .. »

IFTRUE ....

ونرى فى السطر الثالث الأمر IFTRUE يتبعه أمر طباعة الكلمة CORRECT .

ثم نرى فى السطر الرابع نفس الأمر يتبعه الأمر MAKE الذى يستخدم هنا لزيادة رصيد النتيجة بمقدار واحد نظراً لصحة الإجابة .

أى أن الأمر (أو الأوامر) IFTRUE يعقبها كل ما تريد تنفيذه عندما تكون النتيجة صحيحة (TRUE) .

والأمر IFFALSE فى السطر الخامس يعقبه ما نريد تنفيذه عندما لا يتحقق الشرط أى عندما تكون نتيجة المقارنة غير صحيحة (FALSE) . ونرى فى السطر الخامس أنه يتم طباعة الرسالة "....SORRY" يعقبها طباعة الإجابة الصحيحة للسؤال .

والأمر TEST قد لا يتوفر ببعض أجهزة الكومبيوتر لذلك يمكننا بناء نفس منطق هذا البرنامج باستخدام الأمر IF الذى نعرفه فيصبح البرنامج كالآتى :

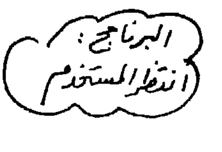
```
TO GETANSWER
MAKE "RESPONSE READNUMBER
IF :RESPONSE = :RIGHTANSWER [PR
"CORRECT MAKE "SCORE :SCORE + 1]
 [PR SE [SORRY, THE ANSWER IS] :
RIGHTANSWER]
END
```

(صورة أخرى للبرنامج GETANSWER)

* بعد ذلك نأتى إلى البرنامج WAITFORUSER بمعنى « انتظر المستخدم » وكل ما يفعله هذا البرنامج هو طباعة الرسالة « اضغط على الزر ENTER » .

وبعد هذه الرسالة يقوم الأمر READLIST بتعليق البرنامج حتى يقرر المستخدم أن ينتقل للسؤال التالى وبالضغط على الزر ENTER فإن الأمر READLIST يستقبل القائمة الخالية [ ] ويطبعها ومع ذلك يمكنك كتابة أى شىء قبل الضغط على الزر ENTER .

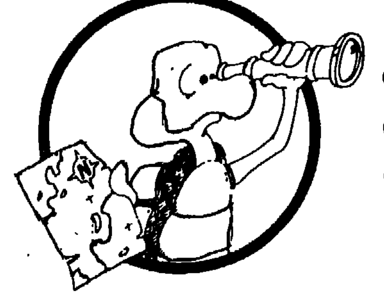
```
TO WAITFORUSER
PRINT [PLEASE PRESS THE ENTER KEY]
PRINT READLIST
END
```



● برنامج النهاية FINISH :

يبدأ البرنامج بمسح شاشة الكتابة . ثم يطبع النتيجة النهائية SCORE متضمنة عدد الأجوبة الصحيحة وعدد الأسئلة الكلى .

تمرين



● يمكنك تعديل البرنامج السابق لكى يقرر اللاعب حدود الأرقام التى سوف يستخدمها فى الامتحان فالبرنامج يمكن أن يوجّه للأطفال الصغار وفى هذه الحالة يمكن أن يحتوى على جمع الأرقام البسيطة التى لا تتعدى العشرة . كما يمكن أن يحتوى البرنامج على عمليات حسابية كبيرة .

● لعله من المناسب أن يحتوى البرنامج على عدّة مستويات من الصعوبة وكلما نجح اللاعب فى أحد المستويات انتقل إلى المستوى الأعلى ..

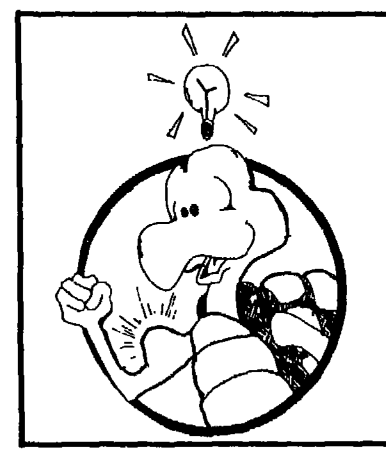
● هل تستطيع أن تطبع رسائل مختلفة عقب كل إجابة سواء فى حالة الإجابة الصحيحة أو الخاطئة ؟

● اكتب التعديل اللازم لتحويل اللعبة إلى امتحان فى الطرح ثم آخر فى القسمة أو الضرب .

● هل يمكنك تعميم البرنامج ليشمل العمليات الحسابية جميعاً ؟

أحد طُرق التنفيذ هى أن يختار اللاعب نوع العملية الحسابية (جمع أو طرح أو ضرب أو قسمة) .





يمكن تطوير اللعبة بطريقة مثيرة للخيال إذا استخدمت السلحفاه فى تقديم المسائل وطباعة النتائج .. إنه مشروع يحتاج إلى التعاون بين مجموعة من الأصدقاء من عشاق لغة لوجو ..

● الكود آسكى لكل زر من الأزرار ASCII

كما نعلم أن كل زر تضغط عليه من لوحة الأزرار ينقل إلى الكومبيوتر رقماً كودّياً معيّناً يمثل اللبنة المكتوبة على الزر .

كما أن بعض أزرار الكومبيوتر لا تكتب شيئاً مثل زر المسافة الخالية وأزرار التحكم الخاصة بإيقاف البرنامج أو الخروج من المحرر .. إلخ .

وجميع هذه الوظائف واللبنات ممثلة بأرقام كودية فى جدول قياسى عالمى يستمى جدول الكود آسكى (ASCII code) وهو موضح بالملحق رقم (٤) .

وهذا الجدول يسمح بتمثيل أكواد مختلفة حتى الرقم 255 ومع ذلك فعدد الأكواد به أقل من ذلك . وهذا يعطى شركات الكومبيوتر الفرصة لابتكار أشكال مختلفة للرسم أو حروف جديدة أو وظائف جديدة للأزرار وتكويدها بداخل الجدول . وبذلك فإن كل كومبيوتر له جدوله الخاص للكود الذى يتضمن الشفرة العالمية علاوة على الشفرات الخاصة له .

ولمعرفة كود أى لبنة من اللبنات المكتوبة على الأزرار نستخدم أمر العملية آسكى كالآتى :

أمر طبع كود الحرف A ← PRINT ASCII ″A

الجواب ← 65

PRINT ASCII ″B

كود الحرف B ← 66

● الموسيقى بلغة لوجو SOUND

تتوفر المؤثرات الصوتية بلغة لوجو بالأجهزة المختلفة بطريقة مشابهة للمؤثرات الصوتية فى لغة بيسك . ومن خصائص الموسيقى بصفة عامة أنها تعتمد كثيراً على طراز الكومبيوتر ، لذلك يوصى بالرجوع إلى كتاب الاستخدام للغة للكومبيوتر المعيّن .

وبصفة عامة فإن أمر الموسيقى عادة يأخذ مُدخلين هما الفترة الزمنية لعزف النغمة وتردد النغمة .

والآتى بعد مثال لأمر الموسيقى بالكومبيوتر المنزلى سنكلير لتوضيح معنى المُدخلات المستخدمة معه :

SOUND [x y]

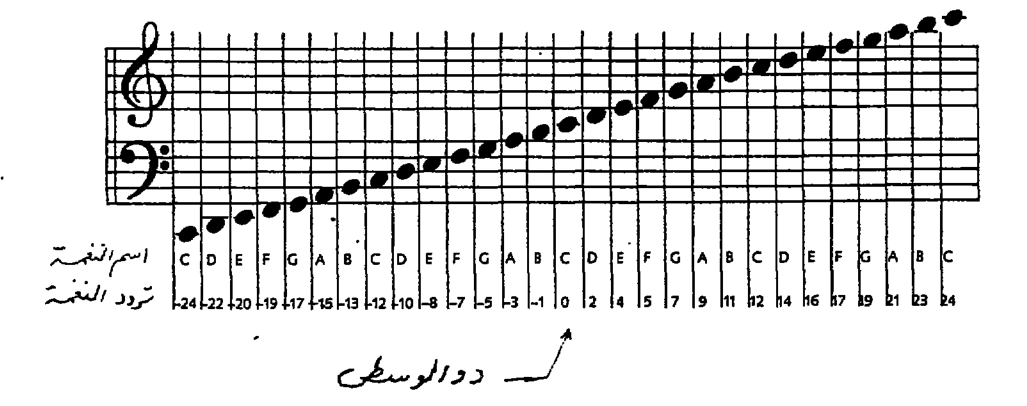
حيث : x

هو زمن النغمة مقاساً بالثوانى ويتراوح بين 10,0

y :

هو النغمة (تردد النغمة) وهى رقم يتراوح بين 60 - ، 69 .

والشكل الآتى يوضح السلم الموسيقى لجهاز سنكلير والقيمة العددية لكل نغمة على مدى ٤ أوكتاف علماً بأن القيمة العددية لنغمة دو الوسطى هى 0 .



والبرنامج التالي يحول أزرار الكومبيوتر سنكلير إلى أزرار بيانو :

```
TO SING
SOUND SE 0.5 (ASCII RC) - 56
SING
END
```

ملاحق الكتاب

الملحق رقم [١] :

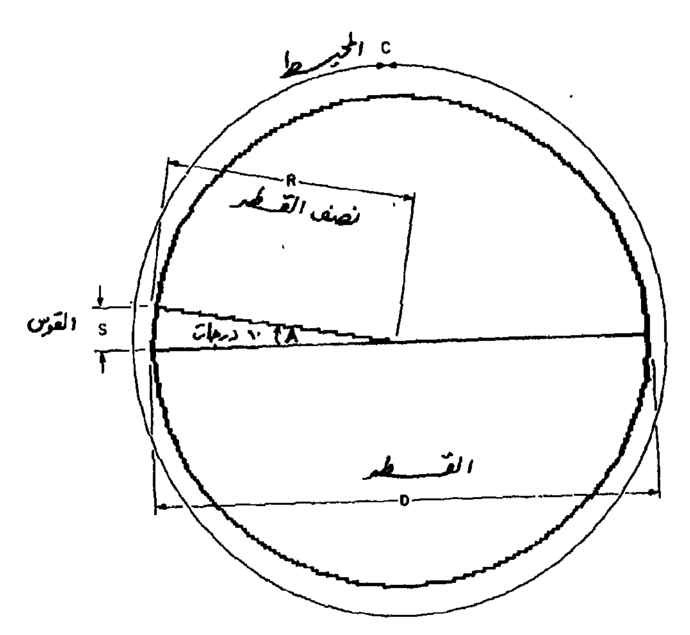
الروتينات الفرعية لرسم دائرة ذات نصف قطر معين R (للكومبيوتر IBM)

نناقش هنا كيفية رسم دائرة ذات نصف قطر معين وهو نفس المبدأ الذى تقوم عليه البرامج الفرعية الجاهزة LCIRCLE, RCIRCLE وأيضاً البرامج الفرعية للأقواس LARC, RARC (للكومبيوتر IBM) وفيما يلى نص هذه البرامج الموجودة على القرص "LWIL" الذى أشرنا إليه من قبل :

```
TO RCIRCLE :R
REPEAT 36 [RCP :R]
END
TO RARC :R
REPEAT 9 [RCP :R]
END
TO RCP :R
RIGHT 5
FORWARD :R * ( 3.14159 ) / 18
RIGHT 5
END
TO LCIRCLE :R
REPEAT 36 [LCP :R]
END
TO LARC :R
REPEAT 9 [LCP :R]
END
TO LCP :R
LEFT 5
FORWARD :R * ( 3.14159 )   18
LEFT 5
END
```

ولنبدأ هنا بمناقشة الروتين RCIRCLE :R لرسم دائرة نصف قطرها R تقع فى الجانب الأيمن من الشاشة . وكما نرى أنه برنامج يعتمد على برنامج فرعى آخر هو RCP :R الذى يستقبل نصف القطر R ويقوم برسم الدائرة باستخدام الأمرين FORWARD , RIGHT .

والدائرة التى نقوم برسمها نقسمها إلى ٣٦ جزءاً أى يتم تقسيم الزاوية ٣٦٠ درجة كلها إلى ٣٦ زاوية صغيرة مقدار كل منها ١٠ درجات . وبذلك فإن طول القوس المواجه لكل ١٠ درجات عبارة عن جزء من ٣٦ جزءاً من محيط الدائرة .



هذا القوس الصغير سوف نعتبره طول الخطوة التى تتحركها السلحفاه أثناء رسمها للدائرة أى أنها سوف تتحرك ٣٦ خطوة وتدور ١٠ درجات فى كل مرة وبذلك تقطع الرحلة الكلية الواقعة فى ٣٦٠ درجة . فما طول القوس S (انظر الرسم) ؟

إن محيط الدائرة هو **C = 2 * 3.14159 * R**

والقوس هو **S = C/36 = 2 * 3.14159 * R/36**

وهذا القوس هو الذى نستخدمه مع الأمر FD باعتباره خطاً مستقيماً (تقريباً) .

FD :R*(3.14159)/18

* ملاحظة [١] :

هذه هى القيم المستخدمة فى البرنامج RCIRCLE للكومبيوتر IBM ولك الحرية أن تقسم الدائرة إلى ما تشاء من الأجزاء بدلاً من ٣٦ جزءاً .

ويمكن اعتبار الصيغة العامة كالآتى :

S = C/N

أى : **S = 2 * 3.14159 * R/N**

حيث N عدد الأجزاء .

* ملاحظة [٢] :

يمكن أيضاً استنتاج طول القوس باستخدام الدوال المثلثية :

S = R * sin A

حيث A زاوية الدوران .

حتى الآن يمكننا كتابة أمر الدائرة كالآتى :

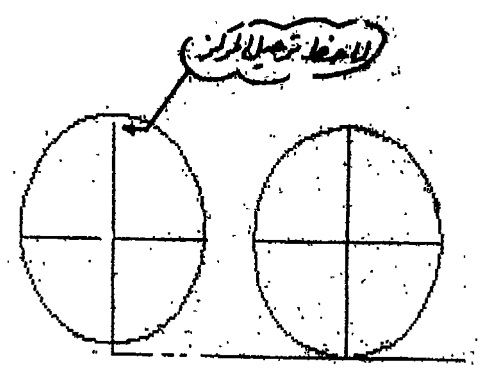
[١] **REPEAT 36 [FD :S RT 10]**

ولكن البرنامج RCP يقسم الزاوية إلى نصفين كالآتى :

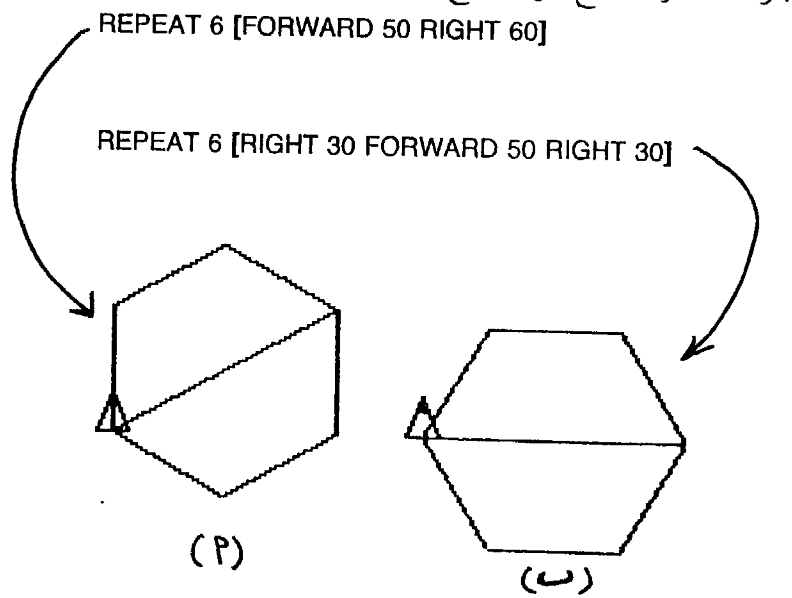
[٢] **REPEAT 36 [RT 5 FD :S RT 5]**

● **وهذه قصة أخرى تتعلق بالدقة :**

فالأمر رقم [١] لا يجعل مركز الدائرة فى المنتصف تماماً بل يجعله أسفل موضعه الأصلى قليلاً ورغم أن هذا لن يكون ملحوظاً مع الدوائر الصغيرة لكنه يظهر كلما زاد حجم الدائرة والآتى بعد شكل يوضح دائرتين إحداهما مرسومة بالأمر [١] والأخرى بالأمر رقم [٢] وتم توصيل خطوط مستقيمة بينهما لتوضيح العيوب المحتملة فى الرسم .



ويظهر هذا العيب جسماً عندما نرسم مضلّعاً ذى عدد قليل من الأضلاع (باعتبار أن الدائرة مضلعٌ ذو أضلاع كثيرة) وجرب المثالين الآتيين :



(أ) (ب)

فبالمثال الأول الذى يرسم الشكل (أ) نلاحظ أن نقطة البداية تقع أسفل مركز الشكل .

أما فى المثال الثانى الذى يعطى الشكل (ب) فتقع نقطة البداية فى المنتصف تماماً .

لذلك تعتبر هذه الملاحظة ملاحظة عامة لكل الأشكال المضلعة .

أما الأقواس RARC ، LARC فهى تعتمد على نفس المبدأ ولكن عدد مرات التكرار تختلف فهى ٩ مرات بدلاً من ٣٦ مرة أى ربع عدد المرات .

الملحق رقم [٢] :

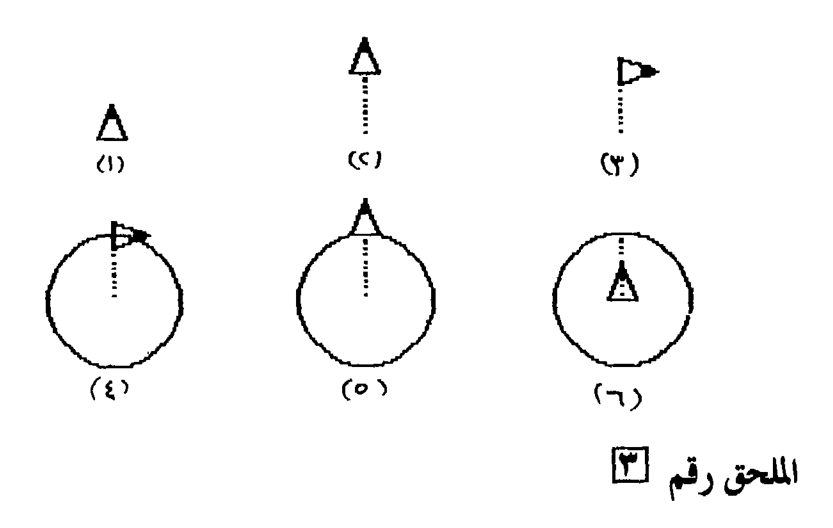
رسم دائرة فى منتصف الشاشة

الروتين الفرعى CCIRCLE

(للكومبيوتر IBM)

أما البرنامج الفرعى لرسم الدائرة فى منتصف الشاشة فهو يتضمن تحريك نقطة البداية إلى أعلى بمقدار نصف القطر R ثم إدارة السلحفاه يميناً بحيث ترسم الدائرة فى الجزء السفلى فتظهر فى منتصف الشاشة وهذا هو الجزء الرئيسى فى البرنامج ويليه خطوات تنفيذ البرنامج :

```
HIDETURTLE
PENUP FORWARD :R ←— (٢)، (١)
RIGHT 90 ←— (٣)
PENDOWN REPEAT 36 [RCP :R] ←— (٤)
LEFT 90 ←(٥)
PENUP BACK :R ←— (٦)
```



الملحق رقم ٣

البرنامج الفرعى لقراءة الأعداد EADNUMBER

(أحد البرامج الفرعية المساعدة على القرص (LWIL procedures) للكومبيوتر IBM)

```
TO READNUMBER
MAKE "NUM1 READLIST
TEST :NUM1 = [ ]
IFTRUE [PRINT [PLEASE TYPE A NUMBER] OUTPUT READNUMBER]
TEST NOT NUMBERP FIRST :NUM1
IFTRUE [PRINT [PLEASE TYPE A NUMBER] OUTPUT READNUMBER]
IFFALSE [OUTPUT FIRST :NUM1]
END
```

● ملاحظة :

يحتوى هذا البرنامج على العملية الشرطية TEST التى ربما لا تتوفر مع بعض الأجهزة وفى هذه الحالة يمكن إنشاء نفس البرنامج الفرعى بطريقة أخرى باستخدام العملية الشرطية IF وحفظة على القرص المغنطيسى بنفس الاسم READNUMBER وقد قدمنا مثل هذا البرنامج فى الباب التاسع .

● فكرة البرنامج :

يستخدم هذا البرنامج لاستقبال الأعداد من لوحة الأزرار بطريقة مماثلة للأمر READLIST الذى يستخدم لاستقبال القوائم .

وللتوضيح جرب المثال الآتى :

```
MAKE "NUMBER READLIST
55
PRINT :NUMBER
55
```

فالعدد 55 تم استقباله من لوحة الأزرار بالأمر READLIST وتمت طباعته أيضاً . وحتى الآن يبدو كل شىء على ما يرام .

ولكن حقيقة الأمر أن هذا العدد ليس إلاّ قائمة بمعنى أنه لا يجوز إدخاله فى أية عمليات حسابية .

وللتأكد من ذلك نجرب المثال الآتى :

```
PRINT (:NUMBER = 55)
FALSE
```

إذن فالعدد 55 ليس رقماً ..

أيضاً عند محاولة جمعه على رقم ما فإن الكومبيوتر يعطى رسالة خطأ ولو أنك كتبت الأمر الآتى :

PRINT :NUMBER + 5

فإنك تستقبل الرسالة التالية من الكومبيوتر :

"+ does'nt like [55] as input"

ولكن يمكن الحصول على العدد 55 من هذه القائمة بطباعة أول كلمة فيها بالأمر الآتى :

PRINT (FIRST :NUMBER)

55

وللتأكد من أن العدد 55 الناتج بهذه الطريقة هو عدد فعلاً أدخل الأمر الآتى لتحصل على النتيجة TRUE :

```
PRINT (FIRST :NUMBER) = 55
TRUE
```

وهدف البرنامج REMDNUMBER هو استقبال العدد وإرساله بالأمر OUTPUT إلى برنامج آخر .

لذلك يمكن كتابة البرنامج ببساطة كالآتى :

```
TO READNUMBER
OUTPUT FIRST READLIST
END
```

هذا هو صلب البرنامج . ويمكن استخدامه بأمر مباشر أو من خلال برنامج . بمعنى أنه طالما يوجد هذا البرنامج معك فى حيز العمل يمكنك إعطاء أمر مثل :

PRINT READNUMBER + 5

عندئذ سوف ينتظر منك البرنامج أن تدخل رقماً ما ثم يجمع عليه الرقم 5 ويطبعه على الشاشة .

أما بقية البرنامج READNUMBER الذى قدمناه بداية فيشمل بعض الشروط التى من شأنها تجنب رسالات الخطأ الناجمة عن سوء الاستخدام وهى الضغط على الزر ENTER بدلاً من إدخال رقم أو كتابة حروف أبجدية .

الملحق رقم ٤

شفرة الكود آسكي القياسية

Character	Code	Character	Code
A	65	0	48
B	66	1	49
C	67	2	50
D	68	3	51
E	69	4	52
F	70	5	53
G	71	6	54
H	72	7	55
I	73	8	56
J	74	9	57
K	75	+	43
L	76	−	45
M	77	/	47
N	78	*	42
O	79	↑	94
P	80	(	40
Q	81	)	41
R	82	<	60
S	83	>	62
T	84	=	61
U	85	?	63
V	86	$	36
W	87	"	34
X	88	,	44
Y	89	.	46
Z	90	,	59
		Carriage Return	13
		Line Feed	10
		Space	32

الملحق رقم ٥ :



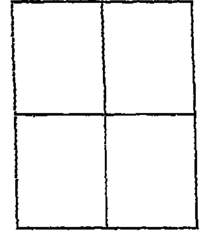
إجابات الباب الرابع

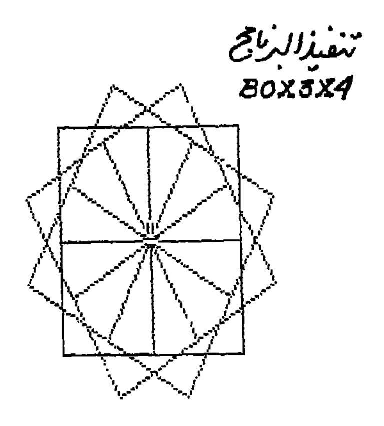
(أ)

```
TO BOX4
REPEAT 4 [REPEAT 4 [fd 40 lt 90]
 RT 90]
END
```

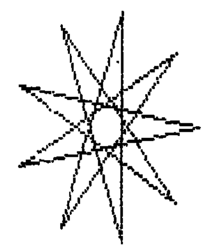
```
TO BOX3X4
REPEAT 3 [box4 rt 30]
END
```

تنفيذ البرنامج BOX4





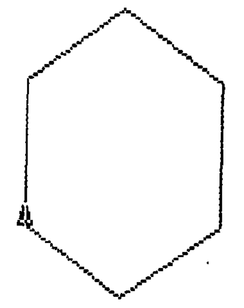
(٢)



```
TO STR9
REPEAT 9 [FD 80 RT ( 360 / 9 ) *
 4]
END
```

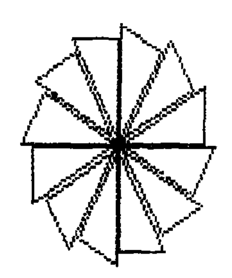
---

(٣)



```
TO POLY6
REPEAT 6 [FD 50 RT ( 360 / 6 ) *
 1]
END
```

(٤)



```
TO SILLY1
HT REPEAT 12 [SILLY]
END
```

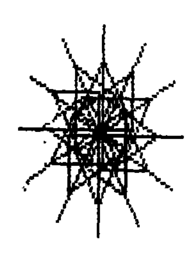
البرنامج الرئيسى

```
TO SILLY
FD 50 LT 90
FD 20 LT 120 FD 30
END
```

البرنامج الفرعى

كما يمكن بالإضافة التالية أن نرسم الشكل التالى :

```
?
TO SILLY2
HT
REPEAT 12 [SILLY RT 60 BK 50]
END
```



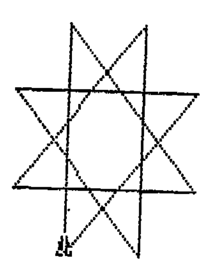
النجمة على اليسار

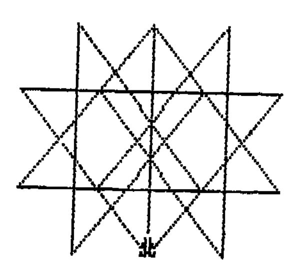
(٥)

```
TO STR8L
REPEAT 8 [FD 80 LT (360/8)*3]
END
```

النجمة على اليمين

```
TO STR8R
REPEAT 8 [FD 80 RT (360/8)*3]
END
```

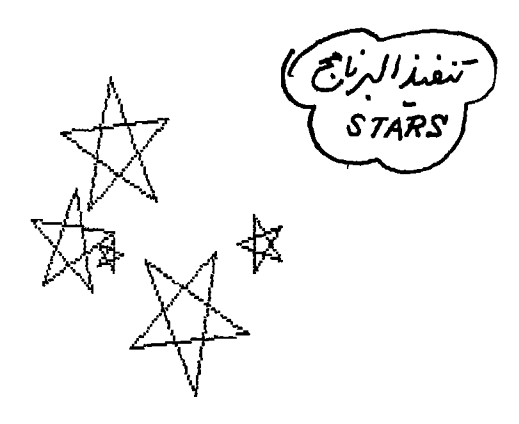




النجمة وصورتها

إجابات الباب السادس

(١)

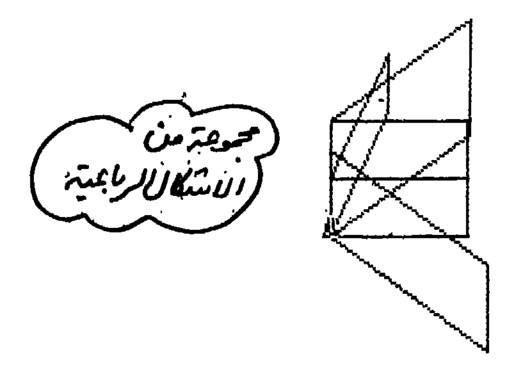



```
?
?
TO STRS
CS HT
REPEAT 5 [STARS 5 + RANDOM 50]
END
```

```
?
TO STARS :SIZE
REPEAT 5 [FD :SIZE RT 144]
PU
LT 60 FD RANDOM 60
PD
END
```

(٢)

```
?
TO RECTG :L :W :ANGLE
REPEAT 2 [FD :W RT :ANGLE FD :L
RT 180 - :ANGLE]
END
```



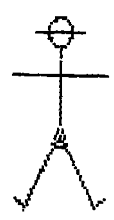
---

(٣)

```
?
TO VAR52
MAKE "AGE 8
MAKE "NAME [SALLY OSSAMA]
MAKE "TEL "65372
MAKE "TELEPHONE :TEL
PR (SE [NAME:] :NAME [AGE:] :AG
E [TEL:] :TEL )
END

NAME  SALLY OSSAMA AGE: 8 TEL: 6
5372
```

إجابات الباب الثامن



بإضافة هذا التعديل إلى البرنامج الفرعى HAT يمكن أن تحصل على الوضع المعدّل للقبعة .

```
?
TO HAT
PU FD 35
PD DRAWHAT
PU BK 40 PD
END
```

التعديل

الملحق [٦] :

ملخص بأوامر لوجو ألتي وردت بالكتاب حسب ترتيب ورودها

الأمـر	الاختصـار
PRINT	PR
CLEARSCREEN	CS
FORWARD	FD
BACK	BK
RIGHT	RT
LEFT	LT
TEXTSCREEN	TS
PENUP	PU
PENDOWN	PD
HIDETURTLE	HT
SHOWTURTLE	ST
SETPAL*	
SETPC	
SETBG	
REPEAT	
FENCE	
WRAP	
WINDOW	
CLEAN	
HOME	
STARTROBOT	
STOPROBOT	
TO	
END	
EDIT	ED
ERASE	ER
ERALL	
PO	

* الأوامر المتبوعة بالعلامة (*) خاصة بالكمبيوتر IBM أو ما يوافقه .

الأمـر	الاختصـار
POALL	
POTS	
SAVE	
LOAD	
DIR*	
CATALOG**	
ERASEFILE*	
SAVEPIC*	
LOADPIC*	
DRIBBLE*	
NODRIBBLE*	
PRINTON**	
RINTOFF**	
COPYSCREEN**	
OUTDEV 1***	
OUTDEV 0***	
RCIRCLE*	
LCIRCLE*	
RARC*	
LARC*	
TRUE	
FALSE	
SUM	
DIV	
PRODUCT	
EQUALP	
ROUND	
INT	
RANDOM	

** أوامر خاصة بكومبيوتر سنكلير
*** أوامر خاصة بكومبيوتر أبل

الأمر	الاختصار
SINE	SIN
COSINE	COS
TANGENT	TAN
ARCSIN	
ARCCOS	
ARCTAN	
COTANGENT	COT
ARCCOT	
REMAINDER	
SQRT	
SENTENCE	SE
WORD	
FIRST	
BUTFIRST	BF
LAST	
BUTLAST	BL
READLIST	RL
MAKE	
SCRUNCH	SCR
SETSCRUNCH	SETSCR
IF	
STOP	
HEADING	
READCHAR	RC
OR	
AND	
OUTPUT	OP
NUMBERP	
NOT	

الأمـر	الاختصـار
TYPE	
CLEARTEXT	CT
READNUMBER*	
TEST	
IFTRUE	
IFFALSE	
ASCII	
SOUND	
CCIRCLE*	

صدر للمؤلف فى مجال الكومبيوتر

من مكتبة ابن سينا

١ ـــ تحدث مع الكومبيوتر بلغة كوبول

... المستوى الأول

٢ ـــ كل شىء عن الكومبيوتر (وكتابة البرامج بلغة بيسك)

... مبسط للنشء ولأولياء الأمور

٣ ـــ تحدث إلى الكومبيوتر بلغة بيسك

... حتى المستوى المتقدم من لغة بيسك يضم اللغة القياسية قديمها وحديثها وأيضاً أشهر طرازات لغة بيسك .

٤ ـــ كيف يفكر الكومبيوتر

... خرائط التسلسل المنطقى للبرامج والنظم الآلية وتحويل النظم اليدوية إلى آلية .

٥ ـــ برمجة الألعاب الكومبيوترية

... طرق برمجة القذائف والتصادم والمؤثرات الصوتية مشروحة بلغة الطرازات الشهيرة للكومبيوتر المنزلى فى مصر والعالم العربى علاوة على لغة بيسك القياسية (ميكروسوفت) .

٦ ـــ مدخلك إلى عالم الكومبيوتر .. المعالجة الإليكترونية للبيانات EDP

٧ ـــ تعلم لغة الكومبيوتر سى من خلال لغة بيسك

مدخل مناسب للهواة والمحترفين لإجادة لغة سى

٨ ــ الرسم بالكومبيوتر

... يتناول كل ما يخص استخدام الكومبيوتر فى الرسم .. يشرح عبارات بيسك القياسية للرسم الدقيق علاوة على أهم اللهجات المنتشرة للأجهزة الكومبيوتر المنزلى .

---

٩ ــ سلسلة قصص الخيال العلمى

١ ــ إعدام إنسان آلى

٢ ــ الدخول فى الثقب الأسود

٣ ــ المعلوم والمجهول

---

١٠ ــ تحدث إلى الكومبيوتر بلغة فورتران ٧٧

... مرجعك العربى فى لغة فورتران يبدا من البدايات الأولى للغة ويصل حتى مستويات متقدمة فى إنشاء البرامج . يضم الكتاب كل عبارات اللغة قديمها وحديثها مع تطبيقات على مختلف أجهزة الكومبيوتر .

١١ ــ برامج وألعاب كومبيوترية مشروحة (بلغة بيسك)

... برامج تعليمية فوازير ألعاب حروب وقذائف ومغامرات .. مقدمة بلغة بيسك على أشهر طرازات الكومبيوتر المنزلى : تكساس ، كومودور ، أتارى ، BBC ، إليكترون ، سنكلير ..

فهرس الكتاب

الموضوع	الصفحة
كلمة المؤلف	٥
■ **الباب الأول : لنكتشف معاً عالم لوجو :**	
• لنتعرف بعائلة لوجو	٩
• إعداد الكومبيوتر للعمل	١٣
• لنكتب أمراً بلغة لوجو PRINT	١٤
• لا تخش أن تتلف الكومبيوتر	١٨
• هيا نلتقى بالسلحفاه CLEARSCREEN, FORWARD, BACK RIGHT, LEFT	
■ **الباب الثانى : التحكم فى السلحفاه :**	
• عالم السلحفاه TEXTSCREEN PENUP, PENDOWN	٢٧
• المزيد من الأوامر للسلحفاه	٢٩
• السلحفاه تحزّم الشاشة	٣١
• لنرسم مربعاً بالسلحفاه	٣٣
• إخفاء السلحفاه وإظهارها HIDETURTLE, SHOWTURTLE	٣٥
• الرسم بالألوان SETPAL, SETPC	٣٧
• تلوين الخلفية SETBG	٣٨
• الألوان وأجهزة الكومبيوتر المختلفة	٣٩
• فلنتقدم خطوة أخرى إلى الأمام	٤٠
• ارسم مربعاً بأمر واحد فقط REPEAT	٤١
• بناء سور حول السلحفاه FENCE, WRAP	
• شاهد السلحفاه من النافذة WINDOW	٤٢
• تنظيف الشاشة بالأمر CLEAN	٤٣
• إعادة السلحفاه إلى بيتها HOME	٤٤
• التحكم فى الروبوط STARTROBOT, STOPROBOT	٤٦

■ الباب الثالث : تعلیم الكومبيوتر أوامر جديدة :

• علّم الكومبيوتر أمراً جديداً TO, END ٤٩
• إصلاح برنامج لوجو EDIT ٥٢
• كتابة البرنامج باستخدام المحرر ERASE, ERALL ٥٤
• ماذا يحدث بداخل الكومبيوتر ؟ ٥٦
• اكتب لغة الكومبيوتر بنفسك ٦٠
• عرض البرنامج على الشاشة PO, POALL, POTS ٦٢
• إمكانات أخرى للمحرر ED, ED [...]
• حفظ البرامج فى الذاكرة الخارجية واستدعاؤها ٦٣
SAVE, LOAD, DIR, ERASEFILE, SAVEPIC, LOADPIC
• تشغيل جهاز الطباعة SAVE LPT1, DRIBBLE, NODRIBBLE
PRINTON, PRINTOFF, OUTDEV ٦٤

■ الباب الرابع : تطبيقات مختلفة للرسم بالسلحفاه :

• استخدام البرامج الفرعية ٦٩
• رسم المثلثات ٧٤
• رسم النجوم ٧٥
• هل هناك بديل للتجربة والخطأ ؟ ٧٨
• المرآة ٧٩
• فكرة لهواة البحث والتنقيب .. فقط ! ٨٠
• الدوائر .. من المبادىء الأولية ٨٤
• ارسم ما تشاء من الأشكال باستخدام الدوائر ٩٠
• المنحنيات ٩٢
• ارسم ثعباناً بالمنحنيات ٩٥
• مشروعات للرسم
١ ــ حدائق الزهور ٩٩
٢ ــ الملامح والوجوه ١٠٤

٣ ــ حيوانات لوجو ١٠٧
٤ ــ ووسائل المواصلات أيضاً ١٠٨
٥ ــ مشروع لعبة كومبيوترية ١١٠

■ الباب الخامس : عالم الأرقام والحروف :

● البيانات فى لغة لوجو ١١٥

● ● التعامل مع الأعداد فى لغة لوجو :

● العمليات الحسابية ١١٦
● المقارنات TRUE, FALSE ١١٧
● هناك دائماً طريقة أخرى PRODUCT, SUM, DIV, EQUALP ١١٩
● التقريب ROUND ١٢٠
● الحذف INT ١٢١
● الأعداد العشوائية RANDOM ١٢٣
● عالم الأعداد فى لغة لوجو ١٢٤
● النسب المثلثية SIN, COS, TAN ....
● النسب المثلثية العكسية ARCSIN, ARCCOS, ARCTAN ....
● مقلوبات النسب المثلثية COT, ARCCOT ....
● باقى القسمة REMAINDER ١٢٥
● إيجاد الجذر التربيعى SQRT ....
● مثال عام : الرسم فى إحداثيات عشوائية ....

● ● التعامل مع الكلمات والقوائم فى لغة لوجو :

● ماهى كلمات لوجو logo words ١٢٦
● تحديد نهاية الكلمة ١٢٨

- القوائم فى لغة لوجو logo lists ١٢٩
- توصيل كلمتين معاً WORD ١٣٠
- توصيل أكثر من كلمتين ..." (WORD ....
- توصيل الكلمات والقوائم فى جملة SENTENCE ....
- توصيل العديد من القوائم والكلمات..(SENTENCE ....
- طباعة جزء من كلمة أو قائمة FIRST, BUTFIRST, LAST, BUTLAST ١٣٢
- الكومبيوتر يسأل وأنت تجيب READLIST ١٣٥
- ماذا يحدث بداخل الكومبيوتر ؟ ١٣٦
- الكومبيوتر يناقش (مثال) ١٤٠
- الكومبيوتر موافق دائماً (مثال) ....

■ **الباب السادس : استخدام المتغيرات :**

- ما معنى المتغيّر MAKE ١٤٥
- اختر الاسم المناسب للمتغير ....
- الكلمات والقوائم فى متغيرات أيضاً ١٤٦
- تخصيص متغير لمتغير آخر ١٤٨
- طباعة العناوين ١٤٩
- الدوال functions ١٥٠
- ارسم مثلثاً متغير الضلع ١٥٢
- نداء الدالة لنفسها recursion ١٥٣
- مازلنا نلعب بالمثلثات المتغيرة ١٥٤
- التكرار مع البرامج أيضاً ١٥٨
- ارسم نجمة خماسية بأحجام مختلفة ١٥٩
- ماذا يحدث بداخل الكومبيوتر ؟ ١٦٠

● الفرق بين المتغيّر العام ومتغيّر الدالة ١٦٤
● الدالة بأكثر من دليل ١٦٦
● تطبيقات ....
١ ــ أشكال زخرفية بالدوائر ١٦٩
٢ ــ رسم دائرة ذات نصف قطر معيّن ١٧١
٣ ــ رسم دائرة فى منتصف الشاشة ١٧٤
٤ ــ رسم مجموعة دوائر ذات مركز واحد ١٧٧
٥ ــ الأشكال الحلزونية ١٧٨
● ضبط الدوائر على الشاشة SCRUNCH, SETSCRUNCH ١٨٠

■ الباب السابع : مشروعات للرسم :

● لنرسم شخصاً ١٨٥
● قافلة من السيارات ٢٠٠
● منطقة سكنية ٢٠٥

■ الباب الثامن : التكرار واتخاذ القرار :

● الحلقات التكرارية ٢١١
● كيف نوقف الحلقة التكرارية IF, STOP ٢١٢
● التحكم فى وضع السلحفاه HEADING ٢١٥
● ماذا يحدث بداخل الكومبيوتر ؟ ٢١٨

■ الباب التاسع : بناء البرامج والألعاب الكومبيوترية :

● تغيير وظائف الأزرار READCHAR ٢٢٩
● الكلمات والقوائم كمتغيرات ....
● ماذا يحدث بداخل الكومبيوتر ؟ ٢٣١

أد ووصل القوائم والكلمات ٢٣٤
يحدث بداخل الكومبيوتر ؟ ....
● استخدام القوائم فى ألعاب الكلمات ٢٤٠
● العمليات المنطقية OR, AND ٢٤٣
● تمرير البيانات من برنامج إلى آخر OUTPUT ٢٤٧
● اختبار الأعداد NUMBERP ٢٤٨
● النفى المنطقى NOT ٢٤٩
● لعبة الرقم السرى TYPE, CLEARTEXT,
READNUMBER ٢٥٠
● لعبة تعليمية : امتحان فى الرياضة TEST, IFTRUE,
IFFALSE ٢٥٩
● الكود آسكى لكل زر من الأزرار ASCII ٢٦٩
● الموسيقى بلغة لوجو SOUND ٢٧٠

■ ملاحق الكتاب :

● الملحق ١

الروتينات الفرعية لرسم دائرة ذات نصف قطر معيّن للكومبيوتر
LCIRCLE, RCIRCLE (IBM)
LARC, RARC ٢٧٢

● الملحق رقم ٢

رسم دائرة فى منتصف الشاشة(للكومبيوتر IBM) CCIRCLE ٢٧٦

● الملحق رقم ٣

البرنامج الفرعى لقراءة الأعداد (للكومبيوتر IBM) READNUMBER ٢٧٧

● **الملحق رقم** ٤

شفرة الكود آسكى القياسية . ٢٨٠

● **الملحق رقم** ٥

إجابات مختارة للتمارين . ٢٨١

● **الملحق رقم** ٦

ملخص بأوامر لوجو التى وردت بهذا الكتاب . ٢٩٠

★★★

رقم الإيداع بدار الكتب ١٩٨٧/٧٩٣٧

الترقيم الدولى ٠ - ٢٠ - ١٣٤١ - ٩٧٧

دار النصر للطباعة الإسلامية
٢ - شارع نشاطى شبرا القاهرة
ت ٧٧٣٢٢١٠

هذا الكتاب ..

يقدم لك لغة « لوجو » التى تحوّل جهاز الكومبيوتر إلى صديق متحدّث ، يحاورك بلغة مفهومة بعيدة عن الرموز والمصطلحات .

إن لغة « لوجو » هى المدخل المناسب للأطفال للتعرّف على الكومبيوتر والتعامل معه وبرمجته . وهى أيضاً لغة المستقبل للصغار والكبار معاً . فهى اللغة الموجهة للتحكم فى الإنسان الآلى (الروبوط) وهى شقيقة لغة ليسب (LISP) لغة الذكاء الصناعى .

إن لغة « لوجو » تستطيع أن تقدم كل ما تقدمه اللغات الأخرى من إمكانات ، ولكن فريقاً من الباحثين المهرة قد بذلوا الجهد الكثير فى تطويرها حتى تصل إلينا فى هذا الثوب السهل الممتنع بعيداً عن تعقيدات الآلة الإليكترونية .

ولغة « لوجو » هى لغة قياسية لم تنتشر فيها اللهجات المختلفة كما فى لغة بيسك . لذلك يسهل تطبيق برامجها على كل أجهزة الكومبيوتر . بل يصل الأمر إلى إمكانية ابتكار أوامر جديدة وإضافتها إلى لغة لوجو لجهازك الخاص .

ومن أهم ما تنفرد به لغة لوجو هى القدرة على أداء الرسومات عالية الدقة بأوامر بسيطة تصدرها إلى السلحفاه البحرية التى تتحرك أمامك على الشاشة . هذا فضلاً عن قدرتها على التلوين وإصدار الأصوات الموسيقية مما يؤهلها لتكون وسيلة فعالة لبرمجة الألعاب الكومبيوترية .

الناشر